M. Y. Hsiao

A Class of Optimal Minimum Odd-weight-column SEC-DED Codes

Abstract: The class of codes described in this paper is used for single-error correction and double-error detection (SEC-DED). It is equivalent to the Hamming SEC-DED code in the sense that for a specified number k of data bits, the same number of check bits r is used. The minimum odd-weight-column code is suitable for applications to computer memories or parallel systems. A computation indicates that this code is better in performance, cost and reliability than are conventional Hamming SEC-DED codes.

Introduction

Single-error-correction, double-error-detection codes (SEC-DED) are widely used to increase computer memory reliability. Examples are error-correction code (ECC) systems for the IBM 7030 and the IBM System/360 Model 85. It has been shown that a memory with ECC, compared to a memory without ECC or to two memories in a duplex configuration, has greatly improved reliability, as judged by performance, cost and size. This improvement is especially evident when the memory system is packaged in a one-bit-per-card base. The new packaging concept was first discussed by Allen,[1] who organized the conventional memory system into the format shown in Fig. 1 so that most error patterns on each card appear as if they were single errors.

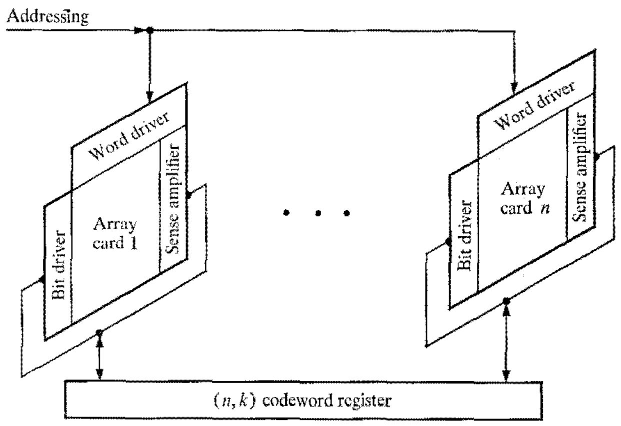

Figure 1 One-bit-per-card organization.

From an error correction point of view, the scheme in Fig. 1 can easily handle errors in the sense amplifier, bit driver, etc. In these cases, each memory cell associated with a given code word is selected independently of all other cells in the same code word. Note that the single large memory has been replaced by a number of smaller submemories, each having an independent set of drive and sensing circuits. This concept facilitates the use of random-error-correcting codes. SEC-DED codes may be practically implemented in this application in contrast to double- or multiple-error-correction codes, which require a greater number of check bits and usually need a more complicated and lengthy decoding process.

The codes described in this paper improve upon the conventional or modified Hamming SEC-DED codes[2] by simplifying the hardware implementation and providing faster and better error-detection capability. The approach used to derive the codes is based on the corollary to Theorem 3.1 stated by Peterson:[3]

"A code that is the null space of a matrix **H** has minimum weight (and hence minimum distance) at least w if and only if every combination of $w - 1$ or fewer columns of H is linearly independent."

Constructing optimal odd-weight column codes

In order to have a SEC-DED code, the minimum weight requirement is 4, which implies that three or fewer columns of the H-matrix are linearly independent. One way to satisfy this condition is to have the columns of the H-matrix meet the following constraints:

1) There are no all-0 columns.
2) Every column is distinct.

The author is located at the IBM Systems Development Division Laboratory, Poughkeepsie, New York.

3) Every column contains an odd number of 1's (hence odd weight).

The first two constraints give a distance-3 code. The third constraint guarantees the code thus generated to have distance 4. The proof of this is simply to consider that the modulo-2 sum of any three odd-weight columns never equals 0. In general, the modulo-2 vector addition of any odd number of odd-weight vectors always gives an odd-weight vector, and any even number of odd-weight vectors gives an even-weight vector including the weight-0 vector. This general statement is actually used for double-error detection. Next, it is realized that the total number of 1's in each row of the **H**-matrix relates to the number of logic levels* necessary to generate the check bit or syndrome bit of that row. Let t_i be the total number of 1's in the ith row, and C_i and S_i be the check bit and syndrome bit specified by the ith row of the **H**-matrix, respectively. Then we have

$$l_{C_i} = \lceil \log_v (t_i - 1) \rceil \tag{1}$$

$$l_{S_i} = \lceil \log_v t_i \rceil , \tag{2}$$

where

$l_{C_i} =$ the number of logic levels required to generate C_i if only a v-input modulo-2 adder is used,

$l_{S_i} =$ the number of logic levels required to generate S_i if only a v-input modulo-2 adder is used,

and $\lceil X \rceil$ is the smallest integer greater than or equal to X. In practical applications, v is fixed for a given circuit family. Therefore, to minimize l_{S_i}, the minimum t_i is desired. If all $t_i (i = 1, 2, \cdots , r)$ are minimum and equal, then we have the fastest encoding and error detection in the decoding process. These are the most critical on-line processes in the memory operations. In general, the code with minimum t_i also requires less hardware for implementation. Less hardware not only implies lower cost but also means better reliability, i.e., if the implementation takes less hardware, it has less chance of failure, since every circuit has an intrinsic failure rate. Therefore, the minimum number of t_i for all i is very important from a practical point of view. The codes constructed by the process suggested in this paper always have fewer 1's in the **H**-matrix than the Hamming SEC-DED codes. In the following section the actual construction procedures are given.

Construction procedures

The construction of the code is best described in terms of the parity-check matrix **H**. The selection of the columns of the **H**-matrix for a given (n, k) code is based on the following three constraints:

1) Every column should have an odd number of 1's; i.e., all column vectors are of odd weight.
2) The total number of 1's in the **H**-matrix should be a minimum.
3) The number of 1's in each row of the **H**-matrix should be made equal, or as close as possible, to the average number, i.e., the total number of 1's in **H** divided by the number of rows.

These constraints obviously result from the reasons stated in the previous section. If **r** parity-check bits are used to match **k** data bits, the following equation must be true:

$$\sum_{\substack{i=1 \\ i=\text{odd}}}^{\leq r} \binom{r}{i} \geq r + k . \tag{3}$$

It can be shown that this code uses the same number of check bits as that of the Hamming SEC-DED code. For an unshortened Hamming SEC-DED code, we have

$$2^{r-1} = k + r , \tag{4}$$

but

$$\sum_{i=0}^{r} \binom{r}{i} = 2^r \tag{5}$$

and

$$\sum_{\substack{i=0 \\ i=\text{even}}}^{\leq r} \binom{r}{i} = \sum_{\substack{i=1 \\ i=\text{odd}}}^{\leq r} \binom{r}{i} ; \tag{6}$$

therefore,

$$\sum_{\substack{i=1 \\ i=\text{odd}}}^{\leq r} \binom{r}{i} = \tfrac{1}{2} 2^r = 2^{r-1} . \tag{7}$$

By comparing Eqs. (4) and (7), one sees that the same number of check bits is required for both codes. The **H**-matrix is constructed as follows:

1) All $\binom{r}{1}$ weight-1 columns are used for the r check-bit positions.
2) Next, if $\binom{r}{3} \geq k$, select k weight-3 columns out of all possible $\binom{r}{3}$ combinations. If $\binom{r}{3} < k$, all $\binom{r}{3}$ weight-3 columns should be selected. The leftover columns are then first selected from among all $\binom{r}{5}$ weight-5 column, etc. The process is continued until all k columns have been specified.

If codeword length $n = k + r$ is exactly equal to

$$\sum_{\substack{i=1 \\ i=\text{odd}}}^{i \leq r} \binom{r}{i}$$

* A logic level is defined as the time required for the signal to pass one transistor gate, e.g., an inverter stage.

Table 1 Sample examples on code parameter relations.

n	k	r	*Structure of* $\mathbf{H}$ $\binom{r}{1}$ $\binom{r}{3}$ $\binom{r}{3}$	*Total number of* 1's *in* $\mathbf{H}$	*Average number of* 1's *in* $\mathbf{H}$ (*for rows*)	$l_{s_i}(\geq l_{c_i})$
12	8	4	$\binom{4}{1}+\binom{4}{3}$	16	4	$\lceil\log_v 4\rceil$
14	9	5	$\binom{5}{1}+9/\binom{5}{3}$*	32	6.4	$\lceil\log_v 7\rceil$
15	10	5	$\binom{5}{1}+\binom{5}{3}$	35	7	$\lceil\log_v 7\rceil$
16	11	5	$\binom{5}{1}+\binom{5}{3}+\binom{5}{5}$	40	8	$\lceil\log_v 8\rceil$
⋮	⋮	⋮				
22	16	6	$\binom{6}{1}+16/\binom{6}{3}$	54	9	$\lceil\log_v 9\rceil$
⋮	⋮	⋮				
26	20	6	$\binom{6}{1}+\binom{6}{3}$	66	11	$\lceil\log_v 11\rceil$
⋮	⋮	⋮				
30	24	6	$\binom{6}{1}+\binom{6}{3}+4/\binom{6}{5}$	86	14.3	$\lceil\log_v 15\rceil$
⋮	⋮	⋮				
39	32	7	$\binom{7}{1}+32/\binom{7}{3}$	103	14.7	$\lceil\log_v 15\rceil$
⋮	⋮	⋮				
43	36	7	$\binom{7}{1}+\binom{7}{3}+1/\binom{7}{5}$	117	16.7	$\lceil\log_v 17\rceil$
⋮	⋮	⋮				
47	40	7	$\binom{7}{1}+\binom{7}{3}+9/\binom{7}{5}$	157	22.4	$\lceil\log_v 23\rceil$
⋮	⋮	⋮				
55	48	7	$\binom{7}{1}+\binom{7}{3}+13/\binom{7}{5}$	177	25.3	$\lceil\log_v 26\rceil$
⋮	⋮	⋮				
72	64	8	$\binom{8}{1}+\binom{8}{3}+8/\binom{8}{5}$	216	27	$\lceil\log_v 27\rceil$
⋮	⋮	⋮				
80	72	8	$\binom{8}{1}+\binom{8}{3}+16/\binom{8}{5}$	256	32	$\lceil\log_v 32\rceil$
⋮	⋮	⋮				
88	80	8	$\binom{8}{1}+\binom{8}{3}+24/\binom{8}{5}$	296	37	$\lceil\log_v 37\rceil$
⋮	⋮	⋮				
96	88	8	$\binom{8}{1}+\binom{8}{3}+32/\binom{8}{5}$	336	42	$\lceil\log_v 42\rceil$
⋮	⋮	⋮				
104	96	8	$\binom{8}{1}+\binom{8}{3}+40/\binom{8}{5}$	376	47	$\lceil\log_v 47\rceil$
⋮	⋮	⋮				
112	104	8	$\binom{8}{1}+\binom{8}{3}+48/\binom{8}{5}$	416	52	$\lceil\log_v 52\rceil$
⋮	⋮	⋮				
120	112	8	$\binom{8}{1}+\binom{8}{3}+\binom{8}{5}$	456	57	$\lceil\log_v 57\rceil$
⋮	⋮	⋮				
128	120	8	$\binom{8}{1}+\binom{8}{3}+\binom{8}{5}+\binom{8}{7}$	512	64	$\lceil\log_v 64\rceil$
130	121	9	$\binom{9}{1}+\binom{9}{3}+37/\binom{9}{5}$	446	49.6	$\lceil\log_v 50\rceil$
⋮	⋮	⋮				
137	128	9	$\binom{9}{1}+\binom{9}{3}+44/\binom{9}{5}$	481	53.4	$\lceil\log_v 54\rceil$
⋮	⋮	⋮				

* The notation $j/\binom{r}{i}$ implies that j out of all possible $\binom{r}{i}$ combinations are used.

$$\mathbf{H}=\begin{array}{c|cccccccccccccccccccccc|c}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & 1 & & & & 1 & & 1 & & & & & & 9\\
2 & 1 & 1 & 1 & & & & 1 & 1 & 1 & 1 & & & 1 & & & & & 1 & & & & & 9\\
3 & 1 & & & & 1 & & 1 & 1 & 1 & & & & & 1 & 1 & 1 & & & 1 & & & & 9\\
4 & & & & & & 1 & 1 & & & 1 & 1 & 1 & & 1 & 1 & 1 & & & & 1 & & & 9\\
5 & & 1 & & 1 & & & & 1 & & 1 & 1 & 1 & 1 & 1 & & & & & & & 1 & & 9\\
6 & & & 1 & 1 & 1 & 1 & & & 1 & & & 1 & 1 & & & 1 & & & & & & 1 & 9\\
 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 4 & 4 & 4 & 2 & 3 & 3 & 3 & & & & & & & \\
 & 2 & 2 & 2 & 6 & 6 & 6 & 3 & 3 & 3 & 5 & 5 & 5 & 5 & 4 & 4 & 4 & & & & & & & \\
 & 3 & 5 & 6 & 5 & 3 & 4 & 4 & 5 & 6 & 2 & 1 & 6 & 6 & 5 & 1 & 6 & & & & & & &
\end{array}$$

Figure 2 H matrix of the (22, 16) SEC-DED code.

for some odd $j \le r$, each row of the H-matrix will have the following number of 1's:

$$\frac{1}{r}\sum_{\substack{i=1\\ i=\text{odd}}}^{i\le r} i\binom{r}{i} = \frac{1}{r}\left[r + 3\,\frac{r(r-1)(r-2)}{3!} + \cdots + j\,\frac{r(r-1)\cdots(r-j+1)}{j!}\right] = \left[1+\binom{r-1}{2}+\cdots+\binom{r-1}{j-1}\right]. \quad (8)$$

If n is not exactly equal to

$$\sum_{\substack{i=1\\ i=\text{odd}}}^{j}\binom{r}{i}$$

for some j, then the arbitrary selection of the $\binom{r}{i}$ cases should make the number of 1's in each row close to the average number, as shown in Table 1.

The double-error detection is accomplished by examining the over-all parity of all syndrome bits. If one has an even number of syndrome bits, then an even number of errors has occurred. Since all errors are assumed to be statistically independent, multiple even errors are treated as if they were double errors. This double-error detection is different from the Hamming code. In the case of Hamming code, a special bit, which is generated by an all-1 row (n 1's) in the **H**-matrix, is examined to determine whether a single (odd) or double (even) error has occurred. The elimination of the all-1 row in the **H**-matrix improves the speed of encoding and decoding for error detection. Another important property of the parity-check matrix, which improves the speed of encoding and decoding for error detection, is the reduced number of 1's in the **H**-matrix, which is always less than in Hamming codes. Moreover, the new **H**-matrix is designed so that $t_i \le \lceil A \rceil$ for all i and $\lceil A \rceil$ (the average number shown in Table 1) is always less than the number of 1's in the row containing the maximum number of 1's in the **H**-matrix of the Hamming SEC-DED code.

Illustrative examples

In this section several parity-check matrices for data lengths 16, 32 and 64 are constructed. The matrices use 6, 7 and 8 check bits, respectively. Figure 2 shows the parity-check matrix for 16 data bits and 6 checks bits. It is constructed according to the $n = 22$ row of Table 1. Note that there are 6 columns corresponding to the 6 possible combinations of 1-out-of-6 and 16 columns corresponding to 16 of the 20 possible combinations of 3-out-of-6. The total number of 1's in the **H**-matrix, therefore, is equal to $6 + (3 \times 16) = 54$ and the average number of 1's in each row is equal to $54/6 = 9$. This implies that if a three-way EXCLUSIVE-OR gate is used ($v = 3$), the check bits and syndrome bits can be generated in two levels. In the conventional Hamming code, three levels are required to use the same kind of EXCLUSIVE-OR gate. Figure 3 shows the hardware layout for the decoder. The encoder can be obtained from the first part of the decoder without using check bits as inputs.

Figure 4 shows the parity-check matrix of the code having 32 data bits and 7 check bits. Figures 5 and 6 show the **H**-matrix of the code for 64 data bits and 8 check bits. In these figures, the eight consecutive 1's of each data byte section are used to generate the byte parity-bit, which requires no additional hardware and is generated before the syndrome bits. These byte parity-bits are usually required when the word is sent to the central processing unit.

Evaluation of the capability of (72, 64) codes

In this section, the capabilities of the modified Hamming (72, 64) code and the (72, 64) codes of Figs. 5 and 6 in this paper are compared. The comparison is based on the assumptions that each code is designed for a binary symmetric channel and that the occurrences of multiple errors are statistically independent. Since the (72, 64) codes are shortened codes and are used for single-error correction and double-error detection, there are cases in which triple errors provide syndrome patterns outside the columns of the code's parity-check matrix. In these cases, the triple error will be correctly detected. If the triple error gives a syndrome pattern coinciding with another column, the decoder is forced to perform a miscorrection. It is important to minimize the probability of miscorrection. The miscorrection probability, denoted by P_3, can be computed as

$$P_3 = \frac{4\,W(4)}{\binom{72}{3}}\,, \quad (9)$$

where $W(4)$ is the number of codewords having weight 4. Equation (9) indicates that among all possible $\binom{72}{3}$ triple error patterns, the number of miscorrection cases is $4W(4)$. Next, among all the possible error patterns, there are

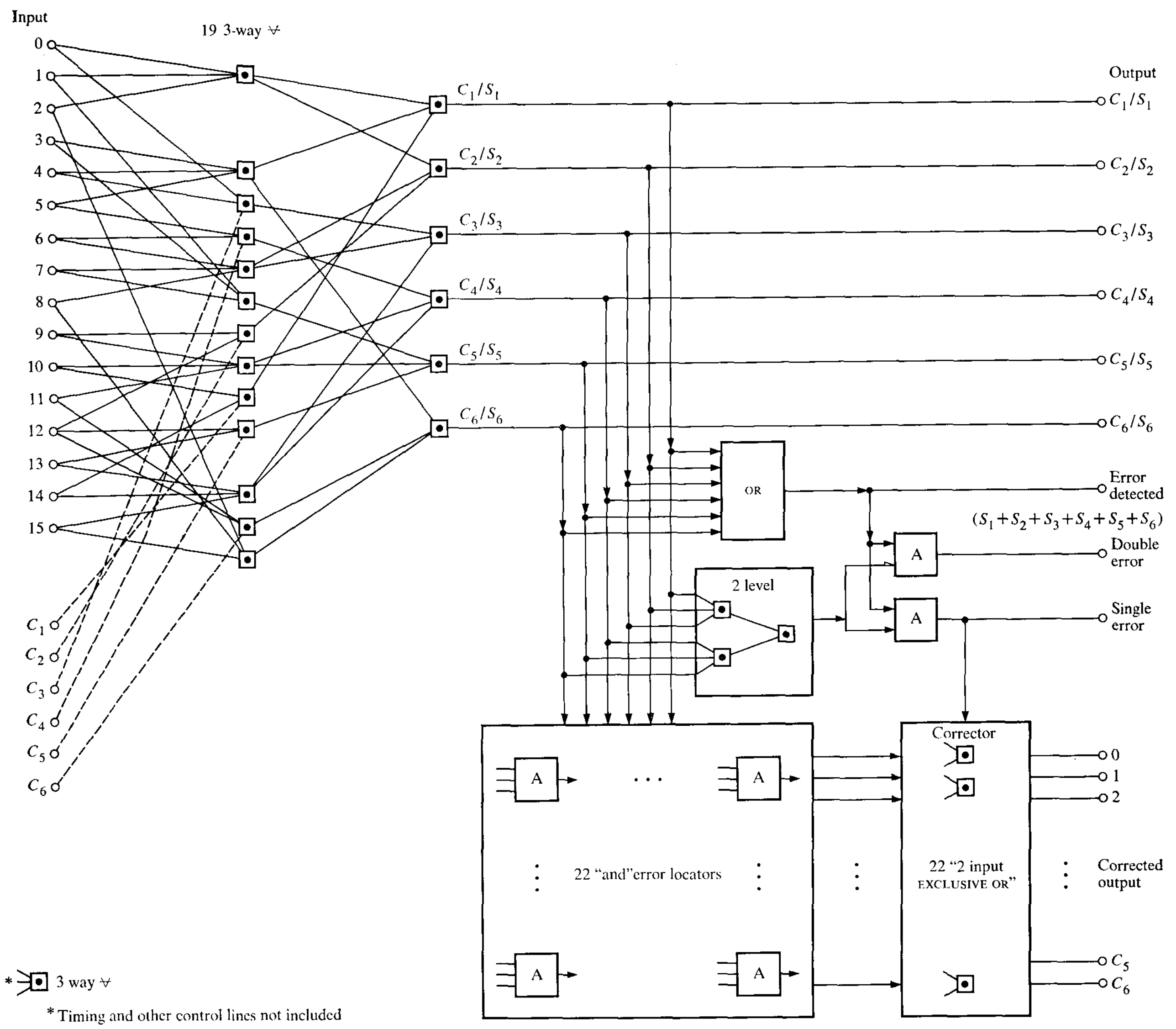

Figure 3 Card layout for ECC encoder and decoder.

Figure 4 Parity-check matrix of the (39, 32) SEC-DED code.

$$\begin{array}{r|ccccccccccccccccccccccccccccccccccccccc|r}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & \\
1 & 1&1&1&1&1&1&1&1&&&&&&&1&&&&&1&&&1&&1&&&&&&1&1&1&&&&&&& 15 \\
2 & &&&&1&&&1&1&1&1&1&1&1&1&1&&&1&&&1&&&1&&&&&1&&&&1&&&&&& 15 \\
3 & &&&1&&&&&&&&1&&&&&1&1&1&1&1&1&1&1&&&1&1&&1&1&&&&1&&&&& 15 \\
\mathbf{H}= 4 & &&1&&&&1&&&&1&&&1&&1&1&&&&&&&&1&1&1&1&1&1&1&1&&&&1&&&& 15 \\
5 & &1&1&&&1&&1&&1&&&1&&&1&&&&&1&1&1&1&&1&1&&1&&&&&&&&1&&& 15 \\
6 & 1&&&&&1&1&&1&&&&1&1&1&&1&1&1&1&1&&&&&&&&1&&&&&&&&&1&& 14 \\
7 & 1&1&&1&1&&&&1&1&1&1&&&&&&1&&&&&&1&&1&&1&&&&1&&&&&&&1& 14 \\
 & 1&1&1&1&1&1&1&1&2&2&2&2&2&2&2&2&3&3&3&3&3&3&3&3&4&4&4&4&4&4&4&4&&&&&&&& \\
 & 6&5&5&3&2&5&4&2&6&5&4&3&5&4&1&4&4&7&2&1&5&2&1&5&1&5&3&3&5&2&1&1&&&&&&&& \\
 & 7&7&4&7&7&6&6&5&7&7&7&7&6&6&6&5&5&6&6&6&6&5&5&7&2&7&5&7&6&3&3&7&&&&&&&&
\end{array}$$

Byte	1								2								3								4								5								6								7								8								Check								
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	
S_1	1	1	1	1	1	1	1	1			1			1	1			1			1			1	1			1								1								1			1	1				1	1	1			1	1	1						1								27
S_2	1	1	1						1	1	1	1	1	1	1	1			1			1	1			1			1			1	1			1								1								1			1	1				1	1	1				1							27
S_3				1	1	1			1	1	1						1	1	1	1	1	1	1	1			1			1	1			1			1			1	1			1								1								1			1	1			1						27
S_4				1			1	1				1	1	1			1	1	1						1	1	1	1	1	1	1	1			1			1	1			1			1			1	1			1								1								1					27
S_5				1								1			1	1				1	1	1			1	1	1						1	1	1	1	1	1	1	1			1			1	1			1			1			1	1			1								1				27
S_6	1			1								1								1			1	1				1	1	1			1	1	1						1	1	1	1	1	1	1	1			1			1	1			1			1			1						1			27
S_7		1			1			1	1			1								1								1			1	1				1	1	1			1	1	1						1	1	1	1	1	1	1	1			1			1	1								1		27
S_8			1			1	1			1			1			1	1			1								1								1			1	1				1	1	1			1	1	1						1	1	1	1	1	1	1	1								1	27

Figure 5 Parity-check matrix of the (72, 64) SEC-DED code, version 1.

Figure 6 Parity-check matrix of the (72, 64) SEC-DED code, version 2.

Byte	1								2								3								4								5								6								7								8								Check							
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
S_1	1	1	1	1	1	1	1	1					1	1	1	1					1	1	1	1					1	1				1	1		1				1				1				1				1				1								1							
S_2	1	1	1	1					1	1	1	1	1	1	1	1									1	1	1	1			1	1		1	1			1				1				1				1				1				1								1						
S_3			1	1					1	1	1	1					1	1	1	1	1	1	1	1					1	1	1	1							1				1				1				1				1				1			1	1				1					
S_4	1	1			1	1	1	1									1	1	1	1					1	1	1	1	1	1	1	1								1				1				1				1				1				1		1	1					1				
S_5		1	1		1				1				1				1				1				1								1	1	1	1	1	1	1	1					1	1	1	1									1	1	1	1			1	1					1			
S_6		1	1			1				1				1				1				1				1							1	1	1	1					1	1	1	1	1	1	1	1					1	1	1	1					1	1								1		
S_7							1				1				1				1				1				1			1	1		1	1			1	1	1	1									1	1	1	1	1	1	1	1					1	1	1	1							1	
S_8								1				1				1				1				1				1		1	1				1	1					1	1	1	1					1	1	1	1					1	1	1	1	1	1	1	1								1

Table 2 Probability of miscorrection and error detection for (72, 64) codes.

Code type	*W*(4)	*Probability of miscorrected triple errors (percent)*	*Probability of detected quadruple errors (percent)*
1) Modified Hamming code	11313	75.88	98.9
2) Fig. 5 code	8392	56.28	99.19
3) Fig. 6 code	8404	56.39	99.18

$W(4)$ cases that will give zero syndrome and thus be undetected. Let P_4 be the probability of undetected quadruple errors, given that four errors have occurred; then

$$P_4 = \frac{W(4)}{\binom{72}{4}}.$$

Since the probability of having one bit in error is low (e.g., of the order of 10^{-5}), we can assume that the probability of having a large number of multiple errors is very small. Two computer programs were used, separately, to find the weight distributions of the three (72, 64) codes. Table 2 compares various P_3 and P_4 for given $W(4)$. Clearly, the codes of Figs. 5 and 6 give better results.

Conclusions

We have demonstrated a new way of contructing a class of SEC-DED codes that uses the same number of check bits as the Hamming SEC-DED code but is superior in cost, performance and reliability. For single-error correction and double-error detection, the class of codes presented here is suitable for computer applications. The condition of having a minimum number of 1's in the rows of the parity-check matrix permits fast generation of check bits and syndrome bits. This rapid generation of bits is important regardless of whether the system has an error or not. Because of the minimum number of 1's, there is a savings in code implementation. This feature also permits minimizing the hardware. Since any hardware circuit has an intrinsic failure rate, however low, a reduction in hardware tends to lower the failure rate of the decoders.

Finally, it was shown that this class of codes has better error detecting capability for triple and quadruple errors than do the conventional modified Hamming codes.

Acknowledgments

I acknowledge the help of many persons in constructing, evaluating and implementing this class of code. Especially, I acknowledge the aid of G. Zook and S. M. Dobrzynski for their construction of the (72, 64) codes; E. Kolankowsky for invaluable help in the implementation; K. G. Smith for a detailed evaluation of the code of Fig. 6, including the value of $W(4)$ for computing the probability of undetected error through a PL/1 program; and T. Takayesu and R. T. Chien for computing the weight distributions of the (72, 64) code of Fig. 5 and the modified Hamming (72, 64) code.

References

1. C. A. Allen, "Design of Digital Memories that Tolerate All Classes of Defects," SEL TR 4662-1, Stanford University, May 1966.
2. R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Tech. J.*, **29**, 147 (1950).
3. W. W. Peterson, *Error Correcting Codes,* M.I.T. Press, Cambridge, Mass. 1961, p. 33.

Received October 21, 1969