From the following examples it will be clear that neither of the above implications can be reversed. Indeed, in two variables over GF(3), $f(x_1, x_2) = x_1 + x_1x_2^2$ is an NSFF but is not an LPP since $f(0, x_2) = 0$. Also $g(x_1, x_2) = x_1^2 + x_2$ is a PP but is not an NSFF over GF(3).

We will now show a particular relation between PP's and NSFF's of degree at most two. By nonsingular linear transformation we mean a transformation of the form

$$x_i = \sum_{j=1}^{n} a_{ij}y_i + b_i \tag{8}$$

where $a_{ij}, b_i \in \mathrm{GF}(q)$ for $1 \le i, j \le n$, and the $n \times n$ matrix $A = (a_{ij})$ is nonsingular so that $\det(a_{ij}) \neq 0$. PP's over GF(q) remain so under a nonsingular linear transformation so that if $f(x_1, \cdots, x_n)$ is a PP and $g(y_1, \cdots, y_n)$ is obtained after a nonsingular transformation of the form (8), then $g(y_1, \cdots, y_n)$ is also a PP. We say that two polynomials are equivalent if one can be obtained from the other by a nonsingular linear transformation of the form of (8). Niederreiter [10] proved the following.

Theorem 4: Suppose $n \ge 2$, and let $f(x_1, \cdots, x_n)$ be a polynomial of degree at most two. If q is odd, then $f(x_1, \cdots, x_n)$ is a PP if and only if $f(x_1, \cdots, x_n)$ is equivalent to a polynomial of the form $g(x_1, \cdots, x_{n-1}) + x_n$ for some polynomial $g(x_1, \cdots, x_{n-1})$. If q is even, then $f(x_1, \cdots, x_n)$ is a PP if and only if $f(x_1, \cdots, x_n)$ is equivalent to $g(x_1, \cdots, x_{n-1}) + x_n$ or $g(x_1, \cdots, x_{n-1}) + x_n^2$ for some polynomial $g(x_1, \cdots, x_{n-1})$.

Clearly, interchanging x_1 and x_n leaves the polynomial $g(x_1, \cdots, x_{n-1}) + x_n$ (or $+ x_n^2$) equivalent to $f(x_1, \cdots, x_n)$. Taking Theorems 1 and 2 into account, Theorem 4 means that if $n \ge 2$ and $f(x_1, \cdots, x_n)$ has degree at most two, then $f(x_1, \cdots, x_n)$ is a PP if and only if $f(x_1, \cdots, x_n)$ is equivalent to an NSFF. Unfortunately, this result does not hold for polynomials of arbitrary degree.

It may be worth pointing out that NSFF's are closely related to combinatorial objects with a "Latin property." For example, every two-stage NSFF over GF(q) corresponds to a unique column Latin square and conversely. A similar statement can be made for n-stage NSFF's. For further details, consult Dénes and Keedwell [1].

In summary, we hope that the theory of PP's and LPP's over finite fields will be of help in the study of NSFF's. Conversely, perhaps the theory of NSFF's can be applied in a fruitful way to the study of PP's and LPP's over finite fields.

Acknowledgment

The author would like to thank S. Suchower for several helpful comments. Thanks are also due the referees for a number of suggestions.

References

[1] J. Dénes and A. D. Keedwell, *Latin Squares and Their Applications*. New York: Academic Press, 1974.

[2] S. W. Golomb, *Shift Register Sequences*, revised ed. Laguna Hills, CA: Aegean Park Press, 1982.

[3] X. Lai, "Condition for the nonsingularity of a feedback shift register over a general finite field," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 5, pp. 747–749, Sept. 1987.

[4] R. Lidl and H. Niederreiter, *Finite Fields, Encyclopedia of Mathematics and Applications*. Reading, MA: Addison-Wesley, 1983, vol. 20 (now distributed by Cambridge Univ. Press).

[5] G. L. Mullen, "Local permutation polynomials over Z_p," *Fibonacci Quart.*, vol. 18, pp. 104–108, 1980.

[6] ——, "Local permutation polynomials in three variables over Z_p," *Fibonacci Quart.*, vol. 18, pp. 208–214, 1980.

[7] ——, "Local permutation polynomials over a finite field," *Norske Vid. Selsk. Skrifter*, no. 1, pp. 1–4, 1981.

[8] ——, "Permutation polynomials in several variables over finite fields," *Acta Arith.*, vol. 31, pp. 107–111, 1976.

[9] H. Niederreiter, "Orthogonal systems of polynomials in finite fields," *Proc. Amer. Math. Soc.*, vol. 28, pp. 415–422, 1971.

[10] ——, "Permutation polynomials in several variables over finite fields," *Proc. Japan Acad.*, vol. 46, pp. 1001–1005, 1970.

[11] C. Ronse, *Feedback Shift Registers*, Lecture Notes in Computer Science. Berlin: Springer-Verlag, 1984, vol. 169.

On the Influence of Coding on the Mean Time to Failure for Degrading Memories with Defects

HAN VINCK AND KAREL POST, MEMBER, IEEE

***Abstract*—We study the application of a combined test-error-correcting procedure to improve the mean time to failure (MTTF) for degrading memory systems with defects. The degradation is characterized by the probability p that within a unit of time a memory cell changes from the operational state to the permanent defect state. We give bounds on the MTTF and show that, for memories with N words of k information bits, coding gives an improvement in MTTF proportional to $(k/n)N^{(d_{\min}-2)/(d_{\min}-1)}$, where $d_{\min}$ and (k/n) are the minimum distance and the efficiency of the code used, respectively. Thus the time gain for a simple minimum-distance-3 code is proportional to $\sqrt{N}$. We also combine a memory word test with a simple defect-matching code. This yields reliable operation with one defect in a word of length $k+2$ at a code efficiency $k/(k+2)$.**

Introduction

In highly dense packed-memory systems we may distinguish between two types of faults; namely, production defects and operational faults [1]. Fault-tolerant techniques can be incorporated in VLSI designs for yield (percentage of good chips) enhancement and performance improvement. In our analysis we assume that the production testing procedure is perfect and that the initial memory system is free of production defects. Then the memory gracefully degrades. The degrading is characterized by the probability p that within a unit of time an *individual* memory cell changes from the operational state to the permanent defect state. This model assumes that the single memory bit cell failure is the dominant failure mode [2]. In the defect state the same value is read from the afflicted cell(s) regardless of what had been written. We distinguish between 0-defects and 1-defects, i.e., between defective cells that always produce a "0" or a "1", respectively, when being read. Furthermore, individual cell failures are isolated and do not affect more than one bit of memory.

Examples of fault-locating procedures can be found in [3], where the whole memory is tested for defects after a period of normal operation. Defects are found by repeated write/read cycles.

We analyze a procedure that uses an error-detecting/correcting code to test whether a memory word contains errors, and thus defective cells, at read time rather than periodic memory testing. It is important to notice that here no errors are detected if the binary contents of a written codeword, together with the stuck-at value of the defects, lead to a valid codeword. Detected errors can possibly be corrected by using erasure decoding, since it is easy to determine the location of defects by a write/read cycle. Hence memory words that are never used or that are without detected errors are never investigated further.

Following [2], we assume that error checking is carried out in parallel with the use of the data. In an error-free state we thus

Manuscript received March 25, 1988; revised October 11, 1988.
The authors are with the Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
IEEE Log Number 8929776.

have no performance change if the error checking is done within a microcycle. However, if errors are to be corrected we need more processing time and also stall/restart capabilities. The average delay caused by the aforementioned procedure is of minor importance as in practice defects are supposed to occur with very small probability.

In the next section we derive lower bounds on the mean time to failure (MTTF) for memories used with the foregoing strategy. In the last section we describe the use of defect matching codes as introduced by Kuznetsov and Tsybakov [4] to tolerate defects.

Mean Time to Failure

In this section we derive two lower bounds for the MTTF for memory systems using erasure decoding of memory words at read time. The results are compared with the MTTF for uncoded memory systems with periodic testing for defects. Before starting the calculations, we specify the model used.

- Each unit of time a specific *word* is selected from the memory. We assume that access frequency is uniform throughout the memory. For a memory with N words the access probability is thus $1/N$.
- The only operational faults that occur are single bit cell faults. These faults are assumed to be independent events. The probability that within a unit of time an *individual* memory cell turns over from the operational state at time $(i-1)$ to the permanent defect state at time i is $p := 1-q$, where p is a constant.
- In the defect state the same value ("0" or "1") is read from the afflicted cell(s) regardless of what had been written.
- A memory *word* consists of k *cells* in the uncoded and n *cells* in the coded situation, respectively.

The Uncoded Situation

We first consider the uncoded situation. Suppose that after each interval of M time units the whole memory is tested for defects. This can be done by repeated read/write cycles. The probability that there are no defects in a memory word of length k at time iM, $i > 0$, given that there are no defects at time 0 is $(1-p)^{iMk}$. The system is said to be in failure if there is a defect in any of the memory *cells*. Thus, the mean time to failure is given by

$$\begin{aligned}\mathrm{MTTF}(N,\text{uncoded}) &= M(1-q^{kNM}) + 2Mq^{kNM}(1-q^{kNM}) \\ &\quad + 3Mq^{2kNM}(1-q^{kNM}) + \cdots \\ &= \frac{M}{1-q^{kNM}} \cong \frac{1}{kNp}\end{aligned}$$

for small p. Note that for the aforementioned uncoded memory we only check the memory after fixed time slots of length M. This is to avoid memory access delay that would be needed if each word read from memory is to be tested (write/read/write cycle).

The Coded Situation

The question arises whether the application of coding [5]–[8] can improve the MTTF. Words of k digits long are encoded as words of length n from an error-correcting code with minimum distance $d_{\min}$. For the calculations we initially assume that each time that we read a word from the memory we immediately write back the complementary word into the memory at the same location. Subsequently, the word is read and compared with the original word. The digits that remain unchanged correspond to defective cells. If the number of defective cells is less than $d_{\min}$, we can certainly recover the original codeword by using erasure decoding (possible because the location of the defects is known). However, the memory must be declared unreliable for further operation if the number of defects in any selected word is greater than or equal to $d_{\min}-1$. If this situation occurs, we can certainly expect uncorrectable defect patterns in the next time slots. Consequently, the mean time to failure can be defined as

$$\begin{aligned}\mathrm{MTTF}(N,\text{coded}) &:= 1\overline{P}_1 + 2P_1\overline{P}_{2|(1)} + 3P_{1|}P_{2|(1)}\overline{P}_{3|(2,1)} + \cdots \\ &= 1 + P_1 + P_1P_{2|(1)} + P_{1|}P_{2|(1)}P_{3|(2,1)} + \cdots\end{aligned}$$

where $\overline{P}_{i|(i-1,i-2,\cdots,1)}$ is the probability that at time i a word is selected with more than $d_{\min}-2$ defects, given that previous selections contained $d_{\min}-2$ or fewer defects. Note that $P_{1|(0)} := P_1$ because initially the memory system is defect free.

For finite memories of size $N > 1$, it is difficult to give a general expression for the MTTF. We give two lower bounds. The first bound is given by the mean time before there is *at least* one memory word in failure. Hence

$$\begin{aligned}\mathrm{MTTF}(N,\text{coded}) &> 1 + P_1^N + P_2^N + P_3^N + \cdots \\ &= \sum_{i=0}^{\infty} P_i^N \qquad (1)\end{aligned}$$

where P_i is the probability that a selected word of length n is not in failure (i.e., it has $d_{\min}-2$ or fewer defects) at time i, and P_i is given by

$$P_i = \sum_{j=0}^{d_{\min}-2} \binom{n}{j} (q^i)^{n-j}(1-q^i)^j.$$

The MTTF$(N,$coded$)$ can also be upper-bounded by

$$\mathrm{MTTF}(N,\text{coded}) < \sum_{i=0}^{\infty} P_i^N + N$$

because it takes on the average N time units before a word that is in failure is found. Consequently, if N is small compared to the MTTF, then (1) is asymptotically tight for small values of p.

Another lower bound is given next. The probability that i consecutive word selections (that are equally probable) contain $d_{\min}-2$ or fewer defects in each individual selection is given by

$$\begin{aligned}P_{(i,i-1,\cdots,1)} &= \frac{1}{N^i}\sum_{\sigma\in S_i} \mathrm{pr}(\sigma \text{ contains } d_{\min}-2 \text{ or fewer defects}) \\ &> P_iP_{i-1}\cdots P_1\end{aligned}$$

where S_i is the set of all N^i possible sequence selections of i words. From this it follows that

$$\mathrm{MTTF}(N,\text{coded}) > \sum_{h=0}^{\infty}\prod_{i=0}^{h} P_i := \mathrm{MTTF}(\infty,\text{coded}). \qquad (2)$$

Note that MTTF$(\infty,$coded$)$ is independent of the number of words in the memory. It can also be interpreted as the MTTF for a memory with infinitely many words.

In the calculations of the MTTF as given in (1) and (2) we used the fact that each word read from memory is checked for defects. However, if we perform error detection by means of the code in use instead of a write/read/write cycle, then the MTTF is certainly increased because undetected defect patterns may occur and thus in this case normal operation continues. Consequently, bounds (1) and (2) can be used as lower bounds in case the write/read/write cycle is replaced by error detection.

In the Appendix we further elaborate on the lower bounds (1) and (2). The result is rather surprising. For a code with minimum distance $d_{\min}$ in combination with the test procedure, the MTTF is lower bounded by

$$\begin{aligned}\mathrm{MTTF}(N,\text{coded}) &> \left\{ N\binom{n}{d_{\min}-1} \right\}^{-1/(d_{\min}-1)} \\ &\quad \cdot \Gamma\left(\frac{d_{\min}}{d_{\min}-1}\right)\frac{1}{|\ln q|} > 0.8\frac{q}{np}N^{-1/(d_{\min}-1)} \qquad (3)\end{aligned}$$

and

$$\text{MTTF}(N,\text{coded}) > \text{MTTF}(\infty,\text{coded})$$
$$> \left\{ \frac{1}{d_{\min}} \binom{n}{d_{\min}-1} \right\}^{-1/d_{\min}} |\ln q|^{-(d_{\min}-1)/d_{\min}} \cdot \Gamma\left(\frac{d_{\min}+1}{d_{\min}} \right) - \frac{1}{2}. \qquad (4)$$

If

$$|\ln q| < \left\{ N^{d_{\min}} \binom{n}{d_{\min}-1} \right\}^{-1/(d_{\min}-1)} \cdot \frac{1}{d_{\min}}.$$

i.e., if p is sufficiently small and $d_{\min} \geq 3$, then bound (3) exceeds bound (4).

For a code with $d_{\min} = 3$, the MTTF(N,coded) for small p is proportional to $(np\sqrt{N})^{-1}$. As another example, one can take a simple single parity check code for which $d_{\min} = 2$, and $n = k+1$. If the parity of a word is not in order, then the memory is declared to be in failure. Otherwise, normal processing goes on. From (3) it follows that

$$\text{MTTF}(N,\text{parity check code}) > \frac{1}{(k+1)N|\ln q|},$$

i.e., we get roughly the same performance as in the uncoded case. The main advantage is that we now have a small encoding delay instead of a test after M time units.

Suppose that after a failure the memory is renewed. Then we are interested in a long time between two failure events. Therefore, we define the *time-gain* factor as

$$\eta = \frac{\text{MTTF}(N,\text{coded})}{\text{MTTF}(N,\text{uncoded})}.$$

By using (3) we obtain

$$\eta > 0.8 \frac{q}{np} N^{-1/(d_{\min}-1)} \frac{(1-q^{kNM})}{M},$$

and if $p < 1/kNM$, then

$$\eta > 0.4 \frac{k}{n} q N^{d_{\min}-2/d_{\min}-1}.$$

For the simplest, and perhaps most practical, situation where $d_{\min} = 3$, we get

$$\eta \cong \frac{k}{n}\sqrt{N}$$

for small values of p. This confirms some of the results obtained in [8].

If, on the other hand, chip surface is costly or the system is unrepairable (satellite systems), then one is interested in the average amount of chip surface needed to realize a time T. The chip *surface gain* is defined as

$$\gamma = \frac{\dfrac{kT}{\text{MTTF}(\text{uncoded})}}{\dfrac{nT}{\text{MTTF}(\text{coded})}} = \frac{k}{n}\eta.$$

Note: If we assume that each individual memory cell follows the exponentional failure law with failure rate λ, then the word reliability function $R(t)$ can be defined as:

$$R(t) = Pr\{\text{an uncorrectable number of defects has not yet occurred in the } i\text{th word at time } t.\}$$
$$= \sum_{j=0}^{d_{\min}-1} \binom{n}{j} (e^{-\lambda t})^{n-j} (1-e^{-\lambda t})^{j}.$$

By using the results of the Appendix, it is easy to lower bound the MTTF of the entire memory system of N words with individual cell failure rate λ as

$$\text{MTTF}(\lambda) = \int_0^\infty R(t)^N \, dt$$
$$> \{ N(d''_{\min}) \}^{-1/d_{\min}} \Gamma\left(\frac{d_{\min}+1}{d_{\min}} \right) \frac{1}{\lambda}. \qquad (5)$$

Defect Matching Code

The application of defect-matching codes [4] can lead to simple test-error-correcting strategies. We give an example.

A message m of *even* length k is encoded as $x = (0, m, P)$, or by the complementary word $y = (1, \overline{m}, \overline{P})$, where P is the parity of the word x. Hence the length of the message is extended to $k+2$. Initially, we only use x to store the messages. Words read from memory are indicated by x' when x is written, or y'' when y' is written. The procedure is as follows:

1) read x' from memory; if the parity is correct, continue else store the complement y' to x' into the memory at the same location;
(2) read y'';
(3) if there is only one defect that causes an error, then the parity of y'' must be correct because y' is the complement to x' and thus the defect must agree with the respective component of y'.

We say that y' matches the defect and, therefore, call this code a "1-defect-matching code." As a consequence of the precoding, x and y'' are complementary, and thus the original message can be recovered. If after some time the same location is read from the memory and no additional defects have occurred, the parity checks and normal operation proceeds. However, *one* additional defect is detected because the parity of x' fails, and processing stops. The memory is in a failure mode when two or more defects occur, thus tolerating one defect in a word.

As an example, suppose that (0,10,1) is stored in memory. If a 1-defect occurs in the third position, we read $x' = (0,1\hat{1},1)$ and the parity fails. Then we store $y' = (1,00,0)$ and read $y'' = (1,0\hat{1},0)$ which gives the correct message 01, in this case equivalent to 10, without a parity failure. Hence if no further defects occurs, y'' is read as a correct word. If, however, one additional defect occurs, the parity fails and two defects are detected. These cannot be corrected by the decoder.

The MTTF can be estimated as before. The performance is the same as for the $d_{\min} = 3$ code. In Fig. 1 we give the MTTF(N,coded) for $N = 4$, 64, and 1024, respectively, and MTTF(∞,coded) together with simulation results. We also indicate the uncoded MTTF for $N = 64$.

Conclusion

We analyze the performance of a combined test-error-correcting procedure for memories with defects. We bound the MTTF for the test method and compare the results with the uncoded situation. We show that coding gives an improvement in MTTF proportional to $(k/n)N^{(d_{\min}-2)/(d_{\min}-1)}$. The improvement in the MTTF for a simple 1-defect-matching test strategy with efficiency $k/(k+2)$ is proportional to $\sqrt{N}$, where N is the size of the memory in terms of the number of words.

Acknowledgment

The authors thank the reviewers for valuable comments and suggestions.

Appendix

We start by proving the following lemma.

Lemma: Let m and l be positive integers, and let the function

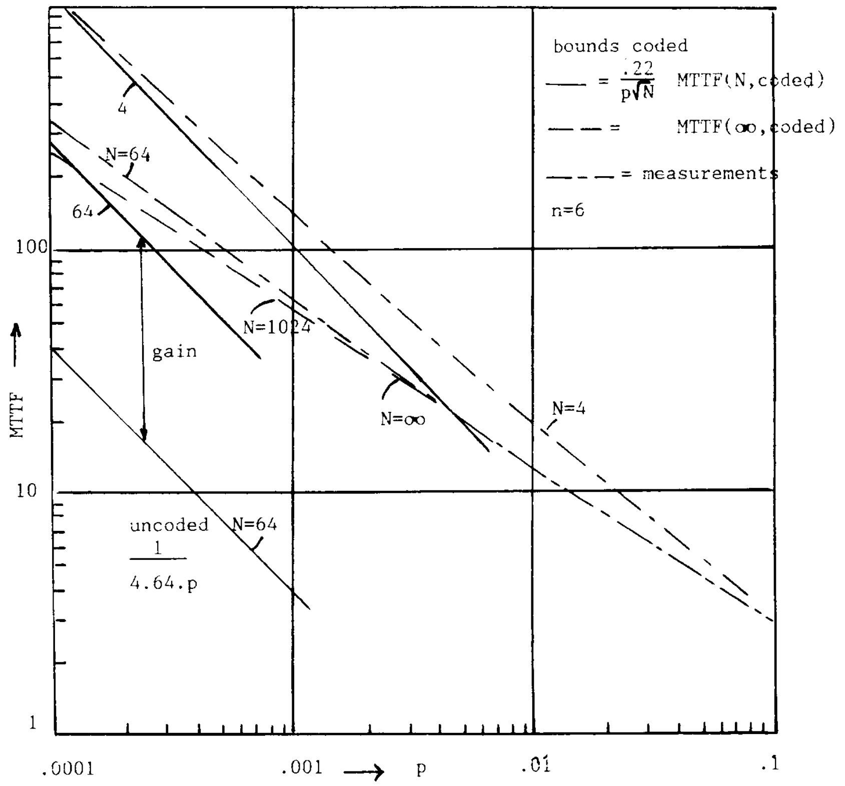

Fig. 1. Mean time to failure for $N = 4$, 64, and 1024 together with measurement results.

$g(v)$ be defined by

$$g(v) := \sum_{k=0}^{l} \binom{m+l}{k} e^{-(m+l-k)v}(1-e^{-v})^k, \qquad v \geq 0.$$

Then

$$g(v) \geq \exp\left(-\binom{m+l}{l+1} v^{l+1}\right), \qquad v \geq 0.$$

Proof: Let

$$f(x) := \sum_{k=0}^{l} \binom{m+l}{k} x^{m+l-k}(1-x)^k.$$

Then $f(0) = 0$ and

$$f'(x) = m\binom{m+l}{l} x^{m-1}(1-x)^l$$

so that

$$\begin{aligned} g(v) = f(e^{-v}) &= m\binom{m+l}{l} \int_0^{e^{-v}} x^{m-1}(1-x)^l\,dx \\ &= m\binom{m+l}{l} \sum_{k=0}^{l} (-1)^k \binom{l}{k} \frac{e^{-(m+k)v}}{m+k} \\ &= m\binom{m+l}{l} \sum_{k=0}^{l} (-1)^k \binom{l}{k} \sum_{j=0}^{\infty} (-1)^j (m+k)^{j-1} \frac{v^j}{j!} \\ &= m\binom{m+l}{l} \sum_{j=0}^{\infty} (-1)^j \frac{v^j}{j!} \sum_{k=0}^{l} (-1)^k \binom{l}{k} (m+k)^{j-1}. \end{aligned}$$

The inner sum in the last expression is the lth order difference $\sum_{k=0}^{l}(-1)^k\binom{l}{k}h(x+k)$ of a function $h(x)$, which is known to be zero if $h(x)$ is a polynomial of degree less than l and has the value $(-1)^l l!$ if $h(x) = x^l$; if $h(x) = x^{-1}$, this lth order difference equals $l!\prod_{k=0}^{l}(x+k)^{-1}$. Hence the power series expansion of $g(v)$ about the point $v = 0$ starts with

$$g(v) = 1 - \binom{m+l}{l+1} v^{l+1} + \cdots.$$

Thus it makes sense to compare the function

$$g(v) = m\binom{m+l}{l} \sum_{k=0}^{l} (-1)^k \binom{l}{k} \frac{e^{-(m+k)v}}{m+k}$$

with the function $\hat{g}$ defined by

$$\hat{g}(v) := \exp\left(-\binom{m+l}{l+1} v^{l+1}\right), \qquad v \geq 0.$$

We verify immediately that $g(0) = \hat{g}(0) = 1$ and that $\lim_{v\to\infty} g(v) = \lim_{v\to\infty} \hat{g}(v) = 0$. Both functions are differentiable, so we study the sign of $(\hat{g}-g)'(v)$ for positive values of v. We claim that there is exactly one positive value of v where it flips (from negative to positive), hence proving the lemma. We observe that for $v \geq 0$,

$$\begin{aligned} \operatorname{sgn}[(\hat{g}-g)'(v)] &= \operatorname{sgn}\left[-v^l \exp\left(-\binom{m+l}{l+1} v^{l+1}\right) + e^{-mv}(1-e^{-v})^l\right] \\ &= \operatorname{sgn}\left[\left(\frac{1-e^{-v}}{v}\right)^l - \exp\left(mv - \binom{m+l}{l+1} v^{l+1}\right)\right] \\ &= \operatorname{sgn}\left[\frac{1-e^{-v}}{v} - \exp\left(\frac{m}{l}v - \frac{1}{l}\binom{m+l}{l+1} v^{l+1}\right)\right] \\ &= \operatorname{sgn}\left[\ln\frac{1-e^{-v}}{v} - \left\{\frac{m}{l}v - \frac{1}{l}\binom{m+l}{l+1} v^{l+1}\right\}\right]. \end{aligned}$$

Now

$$\left(\frac{m}{l}v-\frac{1}{l}\binom{m+l}{l+1}v^{l+1}\right)$$

is a concave function of v that vanishes at $v=0$ and is increasing at $v=0$. On the other hand, $\ln(1-e^{-v})/v$ is a convex function of v that vanishes at $v=0$ and is decreasing at $v=0$. Since the expression between the last brackets is ultimately positive as $v\rightarrow\infty$, the claim, and hence the lemma, is proved.

As a consequence, we can give a lower bound for the probability P_i that a specific word of length $n=m+l$ has no more than l defects at time i as follows:

$$P_i=\sum_{k=0}^{l}\binom{n}{k}(q^i)^{n-k}(1-q^i)^k\ge\exp\left(-\binom{n}{l+1}i^{l+1}|\ln q|^{l+1}\right).$$

Therefore, we obtain for fixed N,

$$\sum_{i=0}^{\infty}P_i^N>\int_0^{\infty}\exp\left(-N\binom{n}{l+1}|\ln q|^{l+1}u^{l+1}\right)du$$
$$=N^{-1/(l+1)}\frac{1}{|\ln q|}\binom{n}{l+1}^{-1/(l+1)}\Gamma\left(\frac{l+2}{l+1}\right).\quad\text{(A1)}$$

On the other hand, since $\sum_{i=0}^{h}i^{l+1}<1/(l+2)(h+(1/2))^{l+2}$, we obtain

$$\sum_{h=0}^{\infty}\prod_{i=0}^{h}P_i>\sum_{h=0}^{\infty}\exp\left(-\frac{1}{l+2}\binom{n}{l+1}|\ln q|^{l+1}\left(h+\frac{1}{2}\right)^{l+2}\right)$$
$$>\int_{1/2}^{\infty}\exp\left(-\frac{1}{l+2}\binom{n}{l+1}|\ln q|^{l+1}u^{l+2}\right)du$$
$$>(l+2)^{1/(l+2)}\binom{n}{l+1}^{-1/(l+2)}$$
$$\cdot|\ln q|^{-(l+1)/(l+2)}\Gamma\left(\frac{l+3}{l+2}\right)-\frac{1}{2}.\quad\text{(A2)}$$

References

[1] I. Koren and D. J. Pradhan, "Modeling the effect of redundancy on Yield and Performance of VLSI systems," *IEEE Trans. Comput.*, vol. C-36, pp. 344–355, Mar. 1987.
[2] S. A. Elkind and D. P. Siewiorek, "Reliability and performance of error-correcting memory and register arrays," *IEEE Trans. Comput.*, vol. C-29, pp. 920–927, Oct. 1980.
[3] W. C. Carter and C. E. McCarthy, "Implementation of an experimental fault-tolerant memory system," *IEEE Trans. Comput.*, vol. C-25, pp. 557–568, June 1976.
[4] A. V. Kuznetsov and B. S. Tsybakov, "Coding for memories with defective cells," *Prob. Peredach. Inform.*, vol. 10, no. 2, pp. 52–60, 1974.
[5] G. C. Clark and J. Bibb Cain, *Error-Correction Coding for Digital Communications*. New York: Plenum, 1981.
[6] C-E W. Sungberg, "Erasure and error decoding for semiconductor memories," *IEEE Trans. Comput.*, vol. C-27, pp. 696–705, Aug. 1987.
[7] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory, applications: A state-of-the art review," *IBM J. Res. Develop.*, vol. 28, pp. 124–134, Mar. 1984.
[8] M. Blaum, R. Goodman, and R. McEliece, "The reliability of single-error protected computer memories," *IEEE Trans. Comput.*, vol. C-38, pp. 114–119, Jan. 1988.

A Bounded-Distance Decoding Algorithm for the Leech Lattice, with Generalizations

G. DAVID FORNEY, JR., FELLOW, IEEE

***Abstract*—An algorithm is given that decodes the Leech lattice with not much more than twice the complexity of soft-decision decoding of the Golay code. The algorithm has the same effective minimum distance as maximum-likelihood decoding and increases the effective error coefficient by less than a factor of two. The algorithm can be recognized as a member of the class of multistage algorithms that are applicable to hierarchical constructions. It is readily generalized to lattices that can be expressed in terms of binary code formulas, and in particular to 'Construction B' lattices.**

I. Introduction

It has long been suggested that codes based on dense lattices could be used on high-SNR band-limited channels to achieve substantial coding gains, in principle approaching channel capacity. The success of trellis codes for just such applications and the recognition that essentially all known good trellis codes can be constructed as 'coset codes' based on lattice partitions (see [1], [2] and the references therein) has had the side effect of refocussing attention on lattices and lattice codes, which are related to trellis codes as block codes are to convolutional codes. Also, advances in microelectronics have made it realistic to consider the 'decoding' of lattices whose structures are quite complex.

The 24-dimensional Leech lattice has become a kind of benchmark for decoding algorithms for a number of reasons. The Leech lattice is the most prominent lattice in lattice theory [3]. It has a high degree of structure that makes it amenable to sophisticated decoding algorithms. Finally, it offers a nominal coding gain of 6 dB [2]. Conway and Sloane [4] devised a maximum-likelihood decoding algorithm requiring approximately 56 000 decoding operations, based on regarding the Leech lattice as the union of 2^{12} cosets of a much simpler lattice. Forney [5] gave an algorithm requiring about 15 000 decoding operations, using a 256-state trellis diagram to represent the lattice. This was improved by Longstaff [6] to about 10 000 operations, using a Wagner-type decoding idea. The Longstaff algorithm has been embodied in a commercial modem [7] with performance comparable to that of the best trellis-coded modems [2]. The latest world record is held by Be'ery *et al.* [8], who have given a maximum-likelihood decoding algorithm that involves about 8000 operations, or, in a further refinement, an average of about 5000 operations (6000 worst case).

In this correspondence we give a suboptimum 'bounded-distance' decoding algorithm that decodes correctly whenever the received word is within the guaranteed error-correction radius of the Leech lattice, so that it has the same 'error exponent' as maximum-likelihood decoding. The 'error coefficient' is shown to increase from 196 560 to 293 712, which implies a performance loss of only about 0.1 dB. The decoding complexity is not much more than that of two soft-decision decodings of the Golay code, which, using the latest algorithms of Be'ery *et al.* [8], is well below 2000 operations.

The decoding algorithm is based on regarding the Leech lattice as the union of two cosets of the Leech half-lattice, which is a lattice with a 'code formula' [2] of the 'Construction B' [10] type.

Manuscript received October 21, 1988; revised November 17, 1988. An earlier version of this paper was published in the Proceedings of the International Workshop on Algebraic and Combinational Coding Theory, Varna, Bulgaria, September 1988.

The author is with Codex Corporation, 20 Cabot Boulevard, Mansfield, MA 02048, USA.

IEEE Log Number 8929777.