AN ABSTRACT OF THE THESIS OF

George A. Converse for the degree of Master of Science in Computer Science presented on December 17, 1985.

Title: Optimal Unidirectional Error-Detecting Codes

Abstract approved: ____________________________

Bella Bose

Various problems related to systematic error-detecting and error-correcting unidirectional codes are discussed. Systematic codes with r check bits are the main topic of the thesis. Classes of codes are presented which work for specific numbers of information bits and then a class of codes is given which detects the same number of unidirectional errors for any number of information bits. This class is shown to be optimal under certain conditions. A result that indicates it is optimal under wider conditions follows. Finally, a class of error correcting codes is presented.
Optimal Unidirectional Error-Detecting Codes

by

George A. Converse

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Completed December 17, 1985
Commencement June 1986
Redacted for Privacy

APPROVED: Associate Professor of Computer Science in charge of major

Redacted for Privacy

Chairman of department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented December 17, 1985

Typed by George Converse for George A. Converse
ACKNOWLEDGEMENT

I would like to thank Professor Bella Bose for his many helpful suggestions during the development of this thesis.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>FIXED LENGTH CODES</td>
<td>6</td>
</tr>
<tr>
<td>MOD M CODES</td>
<td>13</td>
</tr>
<tr>
<td>ERROR CORRECTING CODES</td>
<td>22</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>25</td>
</tr>
</tbody>
</table>
INTRODUCTION

The possibility of error exists in even the most reliable of systems and thus the desirability of detecting and correcting errors exists in any system. As systems become more complex, they increase the likelihood of error even with the most reliable of components. Therefore, it is becoming increasingly important to design data so that errors can be discovered and removed.

A lot of work has already been done in the theory of error detection and correction. Most early work was done in symmetric error control where errors are equally likely to be a bit changing from a 0 to a 1, 0-1 crossovers, or a bit changing from a 1 to a 0, 1-0 crossovers, regardless of other errors. However, there are many situations in which errors are likely to be unidirectional, there may be 0-1 crossovers or 1-0 crossovers but not both in any given block of bits. Although unidirectional error-detecting and correcting codes are likely to be useful in many situations, they have not been investigated as extensively as symmetric errors. The starting point for this thesis is the paper of Bose and Lin.

The rest of this section is devoted to background definitions and some simplifying results. The next section is devoted to a study of error detection in fixed length systematic codes. The third section studies error
detection in systematic codes of arbitrary length. The last section then concludes with a look at a class of (nonsystematic) error correcting codes.

Unless otherwise stated, a code is a set of words of length $n$ over the alphabet $\{0,1\}$. Thus all codes are binary block codes because of the fixed number of 0-1 bits. A code is systematic if the $n$ bits can be divided into $k$ information bits and $n-k$ check bits. An example of a nonsystematic code is the set of all words with exactly $k$ 1's. The Hamming codes, usually presented in terms of parity check matrices, are the standard and most used example of systematic codes. The purpose of the check bits is to detect and/or correct errors in the code word. A code is unidirectional error-detecting (correcting) if it is designed to detect (correct) unidirectional errors. For example, if the "sent" word is 101101, a "received" word of either 111101 or 100001 would have unidirectional errors but a "received" word of 111000 would not.

The Hamming distance $d(x,y)$ between two code words $x$ and $y$ is the number of indices in which the code words differ. For example, the Hamming distance between 110010 and 100111 is 3 since they differ in the second, fourth and sixth places. A code word $x$ is said to cover $y$ if $x$ is 1 whenever $y$ is 1. The word 101110 covers 101000 but 110111 does not. If neither $x$ covers $y$ nor $y$ covers $x$ is true, then $x$ and $y$ are unordered. The basic theorem is the
following result adapted from Bose and Lin.

**THEOREM 1:** A code \( C \) is capable of detecting \( t \) unidirectional errors if and only if for all \( x \) and \( y \) in \( C \) either \( d(x,y) \geq t+1 \) or \( x \) and \( y \) are unordered.

It seems natural to exploit the Basic Theorem by creating codes with unordered words. The following examples will illustrate how this has been done.

**CONSTANT WEIGHT CODES:** These are all codes words with weight (the number of 1's) equal to some constant. They can detect any number of errors because the words are all unordered, but they are not systematic.

**SYSTEMATIC BERGER CODES:** For words with no more than \( 2^r - 1 \) information bits, the \( r \) check bits count, in binary, the number of 0's in the information part. These codes are systematic and they can also detect any number of errors because all words are unordered.

These represent the best possible results for codes which detect all unidirectional errors. For a given number of bits, \( n \), the Constant Weight codes with weight \( \lfloor n/2 \rfloor \) or \( \lceil n/2 \rceil \) yields the maximum number of code words in which all unidirectional errors can be detected. The Systematic Berger Codes do the same if the codes are restricted to systematic codes. With 10 bits available, a constant
weight code with weight 5 would have \( C(10,5) \) or 252 code words and a systematic Berger code with 3 check bits would have \( 2^7 \) or 128 code words.

The Constant Weight mod\((t+1)\) codes below are the best codes for detecting \( t \) unidirectional errors.

**CONSTANT WEIGHT MOD CODES:** Let \( C \) be the set of all words with \( k \mod(t+1) \) 1's. \( C \) is then the constant weight mod\((t+1)\) code and is optimal among all \( t \) unidirectional error-detecting codes with appropriate choice of \( k \). [Borden]

The following Symmetry Principle will allow us to make simplifying assumptions about the form of the code words.

**SYMmetry PRINCIPLE:** Given any code, the code obtained by any permutation of bits is equivalent, i.e. will detect and/or correct the same number and type of errors.

In all the systematic codes in this thesis, the check bit patterns will depend only on the number of 0's among the information bits. Thus the codes can be described by a sequence of check bit patterns where pattern \( i \) is the check bit pattern for all code words with \( i \) 1's in the information bits. For example, consider the code with 3 information bits and one check bit which counts the number of 0's mod 2 in the information part. The code words are then 0001, 0010, 0100, 1000, 0111, 1011, 1101, and 1110.
The corresponding check bit pattern is 1, 0, 1, 0.

In this thesis, I will find it more convenient to describe the check bit patterns than to describe the systematic codes in other ways. Note that if code word x covers code word y, then the check bit pattern for x, $p_i$, covers the check bit pattern for y, $p_j$, and that $i \geq j$. Furthermore, $d(x, y) = d(\text{info } x, \text{info } y) + d(p_i, p_j) = i - j + d(p_i, p_j)$. Thus the number of errors detected by a systematic code is

$$\min[d(p_i, p_j) + i - j - 1].$$

$p_i$ covers $p_j$ and $i > j$.

To continue the example above, the code word 1110 covers the code word 1000; the check bit pattern for 1110 is $p_3 = 0$ and the check bit pattern for 1000 is $p_1 = 0$; finally, $d(1110, 1000) = 3 - 1 + d(0, 0)$ or 2 and the number of errors detected by the code is 1.

Another simplifying principle is the duality principle below.

**Duality Principle:** An equivalent check bit pattern sequence is obtained if 1's are replace by 0's, 0's by 1's, and the sequence is reversed.
FIXED SIZE CODES

In this section we will look at some fixed size codes. These codes will work with a fixed number of information bits. The number of information bits will be at least $2^r$ so these codes can not detect all errors and we will see that the more information bits allowed the fewer the number of errors detected. The first set of codes are the Repeat-One Codes.

REPEAT ONE CODES: The RepeatOne codes are formed by choosing any check bit pattern, not all 0's or all 1's. The sequence of check bit patterns is formed in 3 blocks. The first block is made of all patterns which cover the chosen check bit pattern. The second block is made of all patterns unordered with respect to the chosen pattern. The third block is made of all patterns covered by the chosen pattern. The patterns in each block are arranged in order of decreasing binary value.

For code words with 5 check bits, the following would be an example of a RepeatOne code with the pattern to be repeated being 10101. For less than 33 information bits, the check bit patterns would be ordered according to the number of 0's as follows.
Block 1: 11111, 11101, 10111, and Pattern = 10101.
Block 2: 11110, 11100, 11011, 11001, 11000, 10110, 10011, 10010, 01110, 01101, 01100, 01011, 01010, 01001, 01000, 00111, 00110, 00011, 00010.
Block 3: Pattern = 10101, 10100, 10001, 10000, 00101, 00100, 00001, and 00000.

THEOREM 2: The RepeatOne code will accept $2^r$, $r \geq 2$, where $i$ is the number of 1's in the check bit pattern repeated.

Proof: The number of errors detected is just the number of patterns in block 2. To count the number of patterns in block 2, subtract the number in block 1 and the number in block 3 from $2^r + 1$. The number in block 1 is easily seen to be $2^{r-i}$ and the number in block 3 is clearly $2^i$.

The number of errors detected by a RepeatOne code will be maximum when the weight of the check bit pattern repeated is $\lceil r/2 \rceil$ or $\lfloor r/2 \rfloor$.

Of course, one natural way to allow a longer sequence of check bit patterns and thereby more information bits is to repeat more than one pattern. The next two codes do that but the result depends on whether the two patterns are ordered or unordered. If the pair is ordered, the same number of errors can be detected and more information bits
allowed by including the whole group of check bit patterns covered by the maximum pattern and covering the minimum pattern. This results in the following RepeatGroup codes.

REPEATGROUP CODES: This check-bit pattern sequence is formed in 3 blocks just like the RepeatOne code, but is based on a pair of ordered check bit patterns. Let \( x \) be a check bit pattern, not all 1's, which covers the pattern \( y \), not all 0's. Block 1 is made of all patterns which cover \( y \) and are not covered by \( x \). This is followed by \( x \), all patterns between \( x \) and \( y \), and \( y \). Block 2 is all check bit patterns which are unordered with respect to both \( x \) and \( y \). Then \( x \), all patterns between \( x \) and \( y \), and \( y \) are repeated. Block 3 is then all patterns covered by \( x \) which do not cover \( y \). Again the patterns within each block are arranged in order of decreasing binary value.

An example of a RepeatGroup code follows. The check bit pattern \( x \) is 11100 and \( y \) is 10000.

Block 1: 11111, 11110, 11101, 11011, 11010, 11001, 10111, 10110, 10101, 10011, 10010, and 10001.
RepeatGroup: \( x=11100, 11000, 10100 \), and \( y=10000 \).

Block 2: 01111, 01110, 01101, 01011, 01010, 01001, 00111, 00110, 00101, 00011, 00010, and 00001.
RepeatGroup: 11100, 11000, 10100, and 10000 again.

Block 3: 01100, 01000, 00100, and 00000.
The RepeatGroup code will accept more information bits than the RepeatOne code but it will detect fewer errors. The example above will accept 35 information bits compared to 32 for the RepeatOne example, but it will detect only 14 errors compared to 21 for the RepeatOne example. In the next section, we will look at a code with 5 check bits that will accept any number of information bits but will only detect 11 errors.

**THEOREM 3:** The RepeatGroup code will accept 
\[ 2^{r} + 2^{k-j} - 1 \]
information bits and will detect 
\[ 2^{r} + 2^{k-j} - 2^{r-j} - 2^{k} + k - j \]
errors where \( k \) is the number of l's in the maximum and \( j \) is the number of l's in the minimum element of the group repeated.

**Proof:** Again, count the number of elements in block 1 and block 3. Also count the number of elements in the group and subtract all three from \( 2^{r} \). Finally, to get the number of errors add on the difference between the number of l's in the maximum of the group and the number of l's in the minimum of the group. The number of elements in group 1 is \( 2^{r-j} - 2^{k-j} \). The number of elements in group 3 is \( 2^{k} - 2^{k-j} \). The number of elements in the group is \( 2^{k-j} \). To complete the proof, note that the number of l's in the
maximum is $k$ and the number in the minimum is $j$ so the difference is $k-j$.

It is clear that the number of information bits accepted is determined by the value of $k-j$. The maximum number of errors detected for a given value of $k-j$ is obtained if $r-j$ and $k$ are equal or, if this is not possible, differ by 1. This can be seen if the number of errors detected is rewritten as

$$2^r + 2^{k-j} + k-j - (2^{r-j} + 2^k)$$

where the quantity in parentheses, for a given value of $k-j$, is minimized if $r-j$ and $k$ are as near to equal as possible.

The last fixed sized codes, the RepeatTwo codes, are formed when the two check bit patterns being repeated are unordered.

**RepeatTwo Codes:** Let $x$ and $y$ be two unordered check bit patterns. The sequence of check bit patterns is organized into 5 blocks. Block 1 is all patterns covering $x$. Block 2 is all patterns covering $y$ but not $x$. Block 3 is all patterns unordered with respect to both $x$ and $y$. Block 4 is all patterns covered by $x$ but not $y$. Block 5 is all patterns covered by $y$. Each block is of check bit patterns is arranged in decreasing order. Letting $x = 11100$ and
$y = 00011$, we get the following example of a RepeatTwo Code. Block 1: 11111, 11110, 11101, and 11100.
Block 2: 11011, 10111, 01111, 10011, 01011, 00111, and 00011.
Block 3: 11010, 11001, 10110, 10101, 01110, 01101, 10010, 10001, 01010, 01001, 00110, and 00101.
Block 4: 11100, 11000, 10100, 01100, 10000, 01000, and 00100.
Block 5: 00011, 00010, 00001, and 00000.

The number of errors detected by the RepeatTwo codes is the minimum of the number of patterns between repeating x's and the number of patterns between repeating y's, i.e. the number of elements in block 3 plus the minimum of the number in block 2 and the number in block 4. In the example above, both are equal so the number of errors detected is 7 + 12 or 19. This is the same as the number of errors detected by a RepeatGroup code with x=11100 and y=11000.

**THEOREM 4:** Let x and y be two unordered check bit patterns. Let $i$ be the number of indices where both are 1. Let $j$ be the number of indices where only x is 1 and let $k$ be the number of indices where only y is 1. Then the number of errors detected by the RepeatTwo code for x and y is the minimum of

$$2^r + 2^r - i - j - k - 2^r - i - j - 2^r - i - k - 2^i + k$$

and
$2^r + 2 + 2^i - 2^{r-i-j} - 2^{i+j} - 2^{i+k}$.

Proof: Once again, count the number of elements in each block. Block 1 has $2^{r-i-j}$ patterns. Block 2 has $2^{r-i-k} - 2^{r-i-j-k}$ patterns. Block 4 has $2^{i+j} - 2^i$ patterns. Block 5 has $2^{i+k}$ patterns. This leaves block 3 with

$2^r + 2 - 2^{r-i-j} - (2^{r-i-k} - 2^{r-i-j-k}) - 2^{i+k} - (2^{i+j} - 2^i)$ elements.

The proof is completed by noting again that the number of errors detected is the number of elements in block 3 plus the minimum of the number in block 2 and the number in block 4.

Of course, both the RepeatGroup and RepeatTwo codes could be extended to repeating more elements. However, instead of looking at these, I will move next to the limit and consider the case where every element is repeated as often as needed.
MOD M CODES

**Mod m codes** are codes where the check bit patterns are determined by the number of 0's mod m. As with fixed length codes, the code is determined by a sequence of (m in this case) check bit patterns and I will describe the codes by describing the check bit patterns. The duality principle still holds for the sequence of m check bit patterns. Furthermore, the sequence can begin with any pattern because of its circular nature. The advantage of mod m codes are that a given coding rule will detect the same number of errors for any number of information bits. The price is that the number of errors detected is less than might be detected with a fixed length code for a given number of information bits. The focus will the TwoPart(r,k) codes defined below.

**TWOPART(r,k) CODES:** The first k check bits have a constant weight of k/2. The remaining r-k check bits have, for each possible choice of the first k bits, all words of length r-k. If k is odd, it does not matter whether the first k check bits have weight (k+1)/2 or weight (k-1)/2 so I'll write k/2 in this case too. The words are then arranged in order of decreasing binary value and appended to the information words based on the number of zeros, mod m, in the information part.
The systematic Berger codes above are a special case, the TwoPart(r,r) codes. These codes are also systematic and are designed for the situation where the number of information bits is too great for the systematic Berger codes. These codes will detect up to \((C(k,k/2)-1)2^{r-k} + r - k\) errors. For \(r > 4\), this is maximum if \(k\) is 4 when it is \(5*2^{r-4} + r - 4\). The next table gives two examples of TwoPart(r,k) codes. Even though there are 16 check bit patterns for the TwoPart(5,2) code and 12 for the TwoPart(5,4) code, they are both capable of detecting 11 errors. One nice feature of both of these codes is that they are capable of detecting 11 errors regardless of the number of information bits. This compares with the 14 errors detected by the RepeatGroup code developed in the previous section which could handle 35 information bits.

<table>
<thead>
<tr>
<th>0's mod 16 or 12 respectively</th>
<th>Check Bit Patterns</th>
<th>TwoPart(5,2)</th>
<th>TwoPart(5,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10111</td>
<td>11001</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10110</td>
<td>11000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10101</td>
<td>10101</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10100</td>
<td>10100</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>10011</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10010</td>
<td>10010</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10001</td>
<td>01101</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10000</td>
<td>01100</td>
<td></td>
</tr>
</tbody>
</table>
The purpose of this section is to investigate the optimality of the TwoPart(r,k) Codes. The TwoPart(r,4) codes are shown to be optimal among the single cover codes defined below. Finally, a result indicating that the TwoPart(r,4) code may be optimal under even fewer restrictions is included.

Any mod M code can be considered in a circular fashion to begin with any of its check bit patterns. A check bit pattern in a mod M code will be called a maximum if there are no other check bit patterns in the code which cover it. A check bit pattern will be called a minimum if there are no other check bit patterns in the code which it covers. A mod M code with M different check bit patterns such that each pattern is covered by a unique maximum and covers a unique minimum is called a single cover mod M code.
For a single cover mod M code, the only reasonable way to arrange the code words is to **group** the code words by max-min pair. Each group should begin with the maximum, follow with those check bit patterns between the maximum and the minimum and conclude with the minimum. The patterns between the maximum and minimum can be arranged in any way which guarantees that covering patterns come before covered patterns. One easy way to do this is to arrange them in terms of decreasing value. The number of errors detected, determined by the number of terms in the code and not in the group, is maximized by including in each group all terms between the maximum and minimum. In what follows, all single cover codes will therefore be arranged as above. One result of this last assumption is that each group in a single cover code will have some power of 2 different patterns.

The following notation will be useful in proving the next lemma. Let C be a code specified by a check bit pattern sequence. Let \( C_0 \) be the check bit pattern sequence of the remaining bits for all terms with a 0 in the first column. Let \( C_1 \) be the check bit pattern sequence of the remaining bits for all terms with a 1 in the first column. For example, if C is 111, 110, 011, 100, 001, and 000, then \( C_0 \) will be 11, 01, and 00 while \( C_1 \) will be 11, 10, and 00.
The purpose of the next lemma is to classify all single cover codes with more than $5\times 2^{r-4}$ elements.

**Lemma 1:**

a) The only single cover codes with at least $2^{r-1}$ elements are symmetrically equivalent to the entire code, the TwoPart(r,1) code and the TwoPart(r,2) code.

b) The only other single cover codes with more than $5\times 2^{r-4}$ patterns are symmetrically equivalent to the TwoPart(r,3) code, the TwoPart(r,4) code or a modification of the TwoPart(r,2) code obtained by including one of the groups and a large enough subset of the other.

Proof: The proof actually divides into 2 parts, first we prove part a) assuming that the code has a group at least as large as those in the TwoPart(r,2) code. Then, we prove the rest of part a) and part b) by induction on $r$.

If the largest group in the code has $2^r$ patterns then the code is obviously all elements. If the largest group in the code has $2^{r-1}$ elements, then the code has a group equivalent to the TwoPart(r,1) code. Since there are no check bit patterns which do not cover 000...0, the code is the TwoPart(r,1) code. To complete the first part, if the largest group in the code has $2^{r-2}$ patterns, then it is equivalent to 011...1 to 010...0. Any pattern not covering 010...0 and not covered by 011...1 must begin with 10. Since there are only $2^{r-2}$ patterns beginning with 10 and we need that many more patterns, the code must contain all
patterns beginning with 10 to have $2^{r-1}$ patterns and therefore must be the TwoPart$(r,2)$ code.

The second part is proved by induction on $r$. It is easily seen for $r=4$. For $r>4$, suppose $C$ has more than $5 \cdot 2^{r-4}$ patterns. If $C$ has a group with more than $5 \cdot 2^{r-4}$ elements then $C$ is either the TwoPart$(r,0)$ code or the TwoPart$(r,1)$ code. Thus, consider the two largest groups in $C$. Any two distinct groups in a single cover code have at least two columns in which both groups are constant but different. Let column 0 and column 1 be the two columns (symmetry) for the two largest groups and let the largest group be 0 in column 0 and 1 in column 1. Thus the second largest (maybe as big) will be 1 in column 0 and 0 in column 1. We now consider two single cover codes with $r-1$ check bits derived from $C$. $C_0$ is all check bit patterns resulting from columns 1 to $r-1$ of all patterns in $C$ with a 0 in column 0. $C_1$ is all check bit patterns resulting from columns 1 to $r-1$ of all patterns in $C$ with a 1 in column 0. Either $C_0$ or $C_1$ must have at least $5 \cdot 2^{r-1-4}$ patterns. We will complete the proof assuming that $C_0$ has more than $5 \cdot 2^{r-5}$ patterns. The other case is similar but slightly easier.

The proof examines the various possibilities for $C_0$. If $C_0$ is equivalent to the TwoPart$(r-1,0)$ code then it must come from 011..1 to 000..0. But this would contradict the assumption that $C$ contains 2 groups. If $C_0$ is the
TwoPart(r-1,1) code, then the group C₀ comes from has all 2^{r-2} patterns beginning with 01. Thus C₁ comes from a subset of all patterns beginning with 10 and C is a modification of the TwoPart(r,2) code.

Next suppose that C₀ is a modification of the TwoPart(r-1,2) code. Thus the group C₀ comes from is of the form 0101..1 to 0100..0 and a subset with more than 2^{r-5} elements from 001..1 to 0010..0. However the inductive hypothesis applied to all patterns with 0's in the second column shows that C₁ must be 1001..1 to 1000..0. This has only 2^{r-3} elements. Now we know the subset beginning with 001 has to have more than 2^{r-4} elements and thus must be all those patterns. Thus if C₀ is a modification of the TwoPart(r-1,2) code, C is the TwoPart(r,3) code.

If C₀ is the TwoPart(r,3) code, then it is straightforward to see that C must be the TwoPart(r,4) code. Finally, then let C₀ be the TwoPart(r,4) code. Then C₁ must have more than 2^{r-3} = 4*2^{r-5} elements. But each element in C₁ is in a group with a most 2^{r-5} patterns. Since each pattern mapped into C₁ must begin with 10 or 11 and not cover the patterns 00110, 00101 and 00011 mapped to C₀, there are simply not enough unordered groups to go around.
The main theorem then follows easily by examining the codes with at least $5 \cdot 2^r - 4$ elements and seeing that the best is the TwoPart$(r,4)$ code which detects $5 \cdot 2^r - 4 + r - 4$ errors.

**THEOREM 5:** If $r$ is at least 5, the TwoPart$(r,4)$ code detects as many errors as any single cover mod M code.

The TwoPart$(r,k)$ codes provide local maxima among all mod m codes in the following sense: a 'small' number of additions and/or deletions can not result in a code which detects more unidirectional errors even if the order of the patterns is rearranged.

**THEOREM:** Any addition of a check bit pattern or any deletions of fewer than $C(k,k/2)$ check bit patterns from a TwoPart$(r,k)$ code results in a code which detects fewer errors even if rearrangement of the order of the check bit patterns is allowed.

**Proof:**

Deletions: Until all maxima and minima of the TwoPart$(r,k)$ code are deleted, the number of errors detected is at most the $\#\text{code} - \#\text{group} + \text{weight(large)} - \text{weight(small)}$ for each group where large and small are maximal and minimal elements in the group. Since this number is, for any group that has lost its maximum or minimum but not both, less
than the number of errors detected originally, the new code can not do as well as the TwoPart(r,k) code.

Addition: Any element, a, added to the TwoPart(r,k) code must be covered by at least two of the current maxima or cover at least 2 of the current minima. Say it is covered by two maxima, M1 and M2. The number of errors now detected is no greater than the minima of the following: errors allowed by group 1, errors allowed by group 2, errors allowed by M1 and a, and errors allowed by M2 and a. If the errors allowed by group 1 or the errors allowed by group 2 are decreased then the new code does worse than the TwoPart(r,k) code. If not, where can a be placed. If it comes 'before' M2 then the errors allowed by M2 and a are less than those allowed by the TwoPart(r,k) code. Otherwise, the errors allowed by M1 and a are less than those allowed by the TwoPart(r,k) code.

While the results above can be strengthened to a greater number of terms, the results are not nearly so comprehensible and considered not worth the effort.
ERROR CORRECTING CODES

Of course, error detection is nice but error correction is even better. However, it requires far more bits to correct errors than to detect them. The following definitions and theorem adapted from Bose and Rao are useful. The asymmetric distance between two code words, DA, is the maximum of the number of 1-0 crossovers and the number of 0-1 crossovers. Errors are asymmetric if they are all 0-1 or 1-0 crossovers where the nature of the crossover is known.

THEOREM 6: A binary code with minimum asymmetric distance t+1 is capable of correcting t asymmetric errors.

In this section we will look at a set of error correcting codes suggested by Bella Bose. The following result from R. C. Bose and S. Chowla is used in the description of our Berger-Type Codes.

THEOREM 7: If m = p^n (where p is a prime) and q = (m^{r+1} - 1)/(m - 1) we can find m + 1 integers (less than q) d_0 = 0, d_1 = 1, d_2, ..., d_m such that the sums

\[ d_{i1} + d_{i2} + ... + d_{ir} \]

0 <= i1 <= i2 ... <= ir <= m
are all different mod q.

Letting $m$, $q$, and $d_i$ be as above, define the codes as below.

BERGER-TYPE CODES: Define $F: 2^m \rightarrow \{0, 1, \ldots, q-1\}$ by

$$F(x_1, x_2, \ldots, x_m) = x_1 * d_1 + \ldots + x_m * d_m \mod q.$$  

Then, let $F_i = \{x | F(x) = i\}$. The $F_i$'s are all disjoint and the number of elements in their union is $2^m$. Therefore there must be an $F_i$ with at least $2^m/q$ elements. We chose

RESULT: The Code described above can correct up to r asymmetric errors.

Proof: We need only take $x \neq y$, $x$ and $y$ in $F_i$, and show that $DA(x,y) > r$. Let $i_1, \ldots, i_l$ be indices such that $x_i = 1$ and $y_i = 0$. Let $j_1, \ldots, j_k$ be indices such that $y_j = 1$ and $x_j = 0$.

$$F(x-y) = d_{i_1} + \ldots + d_{i_l} - (d_{j_1} + \ldots + d_{j_k}) = 0.$$  

Now, if $k$ and $l$ are both $\leq r$, we have two different sets of $r$ or fewer indices with the same sum. This can not be, so either $k$ or $l$ is $> r$ and thus so is $DA(x,y)$.

With $p = 2$ and $m = 2^n$, the number of bits used is $m$ and the number of code words achieved is at least $2^m/(1 + m + \ldots + m^r)$. For fixed $r$, this grows almost as fast as $2^m$. Nonetheless, the code is probably not practical. For
example, with n=3 the code produced by the above method involves first discovering a sequence given by the theorem of R. C. Bose and Chowla, e.g. 0, 1, 2, 4, 7, 12, 20, 29, 43. Then it is necessary to discover one of the Fi's with at least 4 elements, e.g. F0 = {00000000, 11010101, 10000011, 11111110}. All this produces a code which does not do as well as the following code discovered by a short inspection, {00000000, 11111111, 11100000, 00000111, 00111100}. Of course, none of these begin to match the C(8,4) or 70 code words available in the Constant Weight code which will detect all unidirectional errors but correct none.

We have seen several examples of codes which either detect or correct unidirectional errors. In the process, we have noted several trade offs. For example, to increase the number of errors detected or corrected, it is necessary to decrease the number of code words or increase the number of bits used. Another trade off is the convenience of a systematic code versus the higher information rate of a nonsystematic code. As we see above, the most dramatic trade off may come when we try to correct errors as well as detect them.
BIBLIOGRAPHY

J. M. Borden, "Optimal Asymmetric Error Detecting Codes", Information Control, to be published.


