# ROBUST ERROR DETECTION IN COMMUNICATION AND COMPUTATIONAL CHANNELS

*Mark G. Karpovsky, Konrad J. Kulikowski, Zhen Wang*

Boston University
Reliable Computing Laboratory
8 Saint Mary's Street, Boston, MA 02215, USA
*(markkar, konkul, lark)@bu.edu*

## ABSTRACT

Classical linear error-detecting codes are not optimum for error detection in communication and computational channel or data compression when the error distributions of a channel are non-stationary or unknown since they do not minimize the worst case error masking probability. Functions with flat autocorrelations can be used to construct optimum codes for such channels. In this paper we present generalized properties as well as optimum constructions for a wide range of practical parameters for such codes. These *robust* codes minimize the worst case error masking probability. We show a range of exemplary applications for which the codes can be used. We demonstrate the possible applications of robust codes for memory devices, data verification, and "lazy" channels.

## 1. INTRODUCTION

Classical linear minimum distance error-detection codes are designed for channels with specific error distritbutions. These codes concentrate their error detection on what are considered the most probable errors which are typically assumed to be those of a small multiplicity. However, in many environments and for many channels the assumptions which make the traditional methods efficient cannot be guranteed. In many channels and environments the error distributions are non-stationary or difficult to predict. Increased scaling of feature size in hardware is causing increased probabilities of multibit faults and a drastic increase in susceptibility to single-event-upsets (SEU) complicating the analysis of erronous behaviors. Additionally, due to the ubiquitis use of mobile computers the environments in which devices operate and communicate change frequently and often drastically which can result in non-stationary error distributions. Making an assumption about the error distributions in such cases can result in a protection with very poor performance if the error distributions change or were predicted incorrectly.

For example, in most complex hardware devices single faults will typically result in errors of various multiplicities at the outputs of the devices depending on the location and type of fault. If such a complex device is protected with a linear 1-dim parity code all errors with even multiplicities will be missed resulting in possibly no protection depending on the actual faults which will occur in the device.

As an alternative to classical linear codes which concentrate their error-detecting power on a certain class of errors, *robust codes* were proposed which provide a uniform or almost uniform error detection against all error patterns and are optimum with respect to the worst case error masking probablility. The nonsystematic robust codes were first introduced in [1] and used for compression of test responses [2]. Constructions of systematic robust codes were proposed in [3] and applied to protect cryptographic hardware against fault attacks [4].

The robust codes, which are designed to provide uniform or almost uniform protection against all errors are based on the functions with optimum or flat autocorrelations. Interestingly, functions with a flat autocorrelation have also been previously used for constructing good minimum distance codes which are designed for specific error distributions [5]. In this paper, we outline the properties and constructions for robust codes and provide several examples of applications where the codes offer an advantage over classical error-detecting codes.

The remainder of the paper is organized as follows. In section 2, we use spectral method to estimate the error detection ability of codes. We show that the best codes in terms of the worst case error masking probability should have an optimum or flat autocorrelation for communication and flat total autocorrelation for compression. In section 3 to 8, we outline the properties, define optimality of robust codes and introduce several constructions of perfect and optimum robust codes. The paper concludes with example applications of the codes to memories, "lazy" channels check-point data verifications.

## 2. AUTOCORRELATIONS AND ERROR DETECTION PROPERTIES OF CODES

Autocorrelations of logic functions are powerful tools for the analysis and synthesis of digital hardware [6]. These functions are analogous to the classical correlation functions employed extensively in telecommunications [7][8], theory of stochastic proceses [9] and are strongly connected to discrete transforms such as Walsh and Vilenkin-Chrestenson transforms [6]. We first review some basic

definitions of autocorrelation functions.

**Definition 2.1** ***(Autocorrelation Function)*** *For function $f : GF(q^k) \rightarrow GF(q^r) : x \rightarrow z = f(x)$, the autocorrelation $B_f(e)$ of $f$ is defined as*

$$B_f(e) = \sum_{x=0}^{q^k-1} f(x)f(x+e), \tag{1}$$

*where $e \in GF(q^k)$, $\sum$ are integer additions and $+$ is addition in $GF(q^k)$.*

It is clear from (1) that $B_f(e)$ is a convolution-type transform of the original function $f$, with the addition of the variable $x$ by $e$ performed in $GF(q^k)$.

Another function which is useful for our later discussion is the *total autocorrelation function*.

**Definition 2.2** ***(Total Autocorrelation Function)*** *For a function $f : GF(q^k) \rightarrow GF(q^r) : x \rightarrow z = f(x)$ let*

$$f_i(x) = \left\{ \begin{array}{ll} 1, & f(x) = i; \\ 0, & f(x) \neq i. \end{array} \right| \, i \in GF(q^r). \tag{2}$$

*Each $f_i$ is a Boolean valued function with autocorrelation $B_{f_i}$. We thereby define the total autocorrelation of $f$ to be*

$$B_{\sum}(e) = \sum_{i=0}^{q^r-1} B_{f_i}(e) \tag{3}$$

$$= \sum_{i=0}^{q^r-1} \sum_{x=0}^{q^k-1} f_i(x)f_i(x+e), \tag{4}$$

*where $e \in GF(q^k)$, $\sum$ are integer additions and $+$ is addition in $GF(q^k)$.*

Autocorrelation functions can be expressed in terms of double Vilenkin-Chrestenson transforms [6].

**Theorem 2.1** *For $f : GF(q^k) \rightarrow GF(q^r)$, denote by $S_f$ its Vilenkin-Chrestenson transform and $S_f^{-1}$ the inverse Vilenkin-Chrestenson transform. Then*

$$B_f(e) = q^k S^{-1}_{(S_f S_f^*)}(e), \tag{5}$$

*where $S_f^*$ is the complex conjugate of $S_f$.*

Theorem 2.1 is a direct analogue to the Wiener-Khinchin theorem in classical Fourier analysis, and for $q = 2$ is called dydaic Wiener-Khinchin theorem. It enables us to use fast algorithms for calculation of spectral transforms to compute autocorrelation functions and simplify our analysis of the error detection ability of codes using spectral methods. These fast algorithms can be found in [6].

In the remaining part of this section, we will establish a relationship between the error detection ability of codes and their autocorrelations. Depending on the applications and the usage of codes, we classify the codes into two categories: codes for communication and codes for compression.

### 2.1. Autocrrelation and Error Detection Codes for Communication and Computational Channels

In communications, data are represented by codewords which are resistent to errors during the transmission and storage. Applications of error detection codes can be found in many different areas such as wireless communication, memory protection, ethernet, and digital audio/video transmission.

**Definition 2.3** ***(Characteristic Function)*** *The characteristic function of a code $C \subseteq GF(q^n)$, is a function $\chi_c : GF(q^n) \rightarrow \{0, 1\}$ defined as*

$$\chi_c(x) = \left\{ \begin{array}{ll} 1, & x \in C \\ 0, & x \notin C \end{array} \right. \tag{6}$$

Characteristic functions of codes are all Boolean valued functions. Their autocorrelations can be caculated as:

$$B_{\chi_c}(e) = \sum_{x=0}^{q^n-1} \chi_c(x)\chi_c(x+e). \tag{7}$$

We will refer to $B_{\chi_c}(e)$ as the autocorrelation of a code $C$.

From (7), we have $B_{\chi_c}(0) = |C|$. Nonzero error $e$ is masked for message $x \in C$ if and only if $\chi_c(x) = \chi_c(x+e) = 1$. Thus $B_{\chi_c}(e), e \neq 0$ is the number of codewords that will mask a given error $e$. Assuming that codewords are equiprobable, the error-masking probability $Q(e)$ for a fixed error $e$ and a code $C$ can be defined as

$$Q(e) = \frac{B_{\chi_c}(e)}{B_{\chi_c}(0)} = \frac{|\{x | x \in C, x+e \in C\}|}{|C|}. \tag{8}$$

In this paper we are mostly interested in constructions and applications of error detection codes for the case where distribution of errors in the channel are unknown or difficult to model. We thereby consider the worst case of error masking probability.

Let $R = \max_{e \neq 0} B_{\chi_c}(e)$. The criterion we select for the designing of codes is: given $n$ and $R$, maximize the number of codewords $|C|$.

From (7),

$$|C|^2 - |C| = \sum_{e \neq 0} B_{\chi_c}(e) \leq R(q^n - 1). \tag{9}$$

Thereby the optimal codes under this criterion are those with a "flat" autocorrelation. That is: $B_{\chi_c}(e) = R$ for all $e \in GF(q^n), e \neq 0$. These codes have uniform protection for all error patterns, and thus gaurantee predictable behavior regardless of the distribution of errors.

### 2.2. Autocorrelation and Error Detection Codes for Compression

Codes can also be used for data compression in applications where the integrity of the original data should be verified. One example is the data verification. Consider two

hosts A and B with messages $M_A$ and $M_B$ respectively. The data verification problem is thus to determine the minimum amount of information (called signature) needed to be exchanged betwen A and B to verify whether $M_A = M_B$ and guarantee the required error detection ability. Another example is the compression of the test response in hardware testing, where we need a signature that is as short as possible to verify the functionality of the device.

For these applications, the original data and the compressed signature can be treated together as a codeword of a systematic code. The error detection ability of codes should be evaluated according to its protection against errors occuring in the information parts.

Let $f : GF(q^k) \rightarrow GF(q^r)$ be the encoding function of the systematic code used for compression. The total autocorrelation of the encoding function

$$B_{\sum}(e) = |\{x|f(x) = f(x+e)\}|. \tag{10}$$

is the number of codewords that will mask nonzero error $e \in GF(q^k)$ which affect the information part of the codewords. Like in communications, our goal in robust compression is to reduce the worst case error masking probability and so the best codes should have "flat" total autocorrelation functions.

## 3. DEFINITION OF ROBUST CODES

**Definition 3.1** ***(R-Robust Code)*** *A code $C \subseteq GF(q^n)$ is* ***R-robust*** *if the size of intersection of the code $C$ and any of its translates $\tilde{C}_e = \{\tilde{x}|\tilde{x} = x+e, x \in C, e \in GF(q^n)\}$ where $0 \neq e \in GF(q^n)$ is upperbounded by $R$:*

$$R = \max_{0 \neq e \in GF(q^n)} |\{x|x \in C, x+e \in C\}|, \tag{11}$$

*where $+$ is addition in $GF(q^n)$. A $q$-ary $R$-robust code $C$ of length $n$ with $M = |C|$ is denoted by a triple $(n, M, R)_q$. ( In this paper we denote by $[n, M, d]_q$ a $q$-ary linear code with length $n$, $M = |C|$ and hamming distance $d$.)*

A graphic depiction of the definition of a robust code is shown in Figure 1. Let $C \subseteq GF(q^n)$, and $\tilde{C}_e$ be the set of all codewords of $C$ shifted by an element $e \in GF(q^n)$. The code C is **R-robust** if for any $0 \neq e \in GF(q^n)$, the size of the intersection of the two sets C and $\tilde{C}_e$ is upperbounded by $R$.

An alternative definition of robust codes can be based on the autocorrelation of the code.

**Definition 3.2** *A code $C \subseteq GF(q^n)$ is a $R$-robust code iff the autocorrelation of the characteristic function of the code, $B_{\chi_c}(e)$ is bounded by $R$ for any $e \neq 0$.*

$$R = \max_{0 \neq e \in GF(q^n)} \sum_{x=0}^{q^n-1} \chi_c(x)\chi_c(x+e). \tag{12}$$

The above defined robust codes have beneficial properties when worst case error masking probability of the codes is considered. By definition of an $R$-robust code

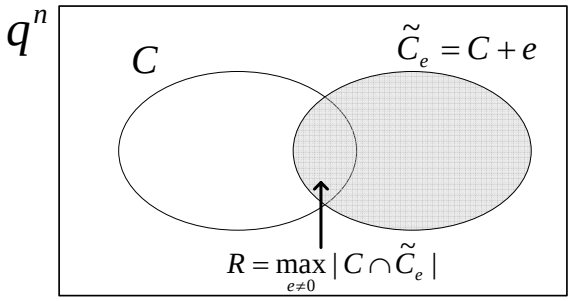



Figure 1. Definition of Robustness

there are at most $R$ codewords which can mask any fixed error $e$. The worst case probability of masking an error for a $(n, M, R)_q$ robust code when the codewords are assumed equiprobable is at most $R/M$. Thereby, robust codes have a predictable behavior in the presence of unpredictable error distributions as the worst case probability of masking of any error is bounded. We next investigate the optimality of robust codes .

## 4. ROBUST CODES WITH FLAT AUTOCORRELATIONS FOR ERROR DETECTION

Based on the above definitions of the robust codes it is possible to derive the following main property for a R-robust code.

**Property 4.1** *If the code $C$ is R-robust, then in the multiset $S_C = \{\alpha - \beta|\alpha, \beta \in C, \alpha \neq \beta\}$ any element appears at most $R$ times.*

For the design of robust codes good robust codes are those achieving the maximum possible $M$ for given $R$ and $n$. From Property 4.1, a relation on $R$, $n$ and $M$ of the code can be established.

$$M^2 - M \leq R(q^n - 1). \tag{13}$$

A robust code which satisfies the equality in (13) has the largest possible number of codewords for given $R$ and $n$. For such a code, $B_{\chi_c}(e)$ (thus $Q(e)$) is totally flat and each nonzero element from $GF(q^n)$ appears exactly $R$ times in multiset $S_C$. Such codes are referred to as *perfect*.

**Definition 4.1** ***(Perfect Robust Code)*** *A robust $(n, M, R)_q$ code satisfying*

$$M^2 - M = R(q^n - 1) \tag{14}$$

*is* ***perfect****.*

Perfect robust codes correspond to extensively studied combinatorial structures known as difference sets and symmetric designs [10].

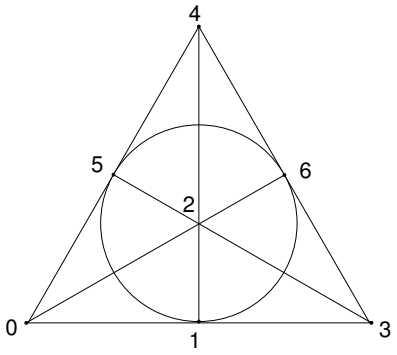



Figure 2. Projective plane of order 2

**Definition 4.2** ***(Difference Set)*** *Let G be a group of order $v$, and $C$ a M-subset of G. Then C is called a $(v, M, R)$-difference set if the list of differences $S = (\alpha - \beta : \alpha, \beta \in C, \alpha \neq \beta)$ contains each nonzero element of G exactly R times.*

Clearly, $(q^n, M, R)$ difference sets in $GF(q^n)$ are exactly perfect $(n, M, R)_q$ robust codes.

Despite the extensive research of the combinatorial structures it is still not known in the general case for what parameters such difference sets and hence perfect robust codes exist, and we note that the perfect robust codes based on the known difference sets have a high complexity of decoding (detecting for a given $x$ whether $x \in C$ or $x \notin C$). A good summary of existing difference sets can be found in [11].

**Example 4.1** *The projective geometry $PG(m, q)$ [12] is an example of a classic combinatorial balanced-imcomplete design [11] that generates a difference set with parameters $(v = \frac{q^{m+1}-1}{q-1}, M = \frac{q^m-1}{q-1}, R = \frac{q^{m-1}-1}{q-1})$, where q is a prime power and $m > 1$ is an integer. These are known as **Singer difference sets**. They are $(1, \frac{q^m-1}{q-1}, \frac{q^{m-1}-1}{q-1})_{\frac{q^{m+1}-1}{q-1}}$ perfect robust codes.*

*$PG(2, 2)$ is shown in Figure (2). The set of any collinear points forms a perfect $(1, 3, 1)_7$ robust code. The differences of the three codewords cover all nonzero elements of $GF(7)$ exactly once.*

Parameters of several perfect nonbinary codes based on difference sets are given in Table 3.

It has been shown that all difference sets over binary fields, thus binary robust codes which are the most important and practical for hardware design, exist only for even dimensions and have the following parameters: $(2s, 2^{2s-1} \pm 2^{s-1}, 2^{2s-2} \pm 2^{s-1})_2$ [11]. These codes can be constructed using the method presented in [1].

Systematic codes, which are often more practical for many applications due to their seperation of data and check bits, cannot be perfect.

**Theorem 4.1** *For any $(n, q^k, R)_q$ systematic robust code there are at least $q^{n-k}$ elements which cannot be expressed as differences of two codewords. .*

**Proof** For any systematic codeword $x = (x_1, x_2)$ an error $e = (e_1, e_2)$ is maksed iff $f(x_1 + e_1) = x_2 + e_2$, $x_1, e_1 \in GF(q^k)$, $x_2, e_2 \in GF(q^{n-k})$. Clearly an error $e = (e_1 = 0, e_2 \neq 0)$ is never masked since $f(x_1) = x_2 + e_2$ only if $e_2 = 0$. An error that is never masked cannot be expressed as a difference of two codewords. Hence elements from $GF(q^n)$ of the form $e = (0, e_2 \in GF(q^r))$ cannot be expressed as a difference of two codwords. □

**Corollary 4.1** *There are no perfect systematic robust codes.*

For most practical parameters perfect robust codes do not exist. When perfect robust codes are not available, the best possible codes containing maximum M for given R and n are referred to as *optimum* robust codes.

**Definition 4.3** ***(Optimum Robust Code)*** *A $(n, M, R)_q$ robust code which has the maximum possible number of codewords $M$ for a given length $n$ and robustness $R$ with respect to (13) is called **optimum**. For optimum codes adding any additional codewords would violate the bound (13) thus*

$$M^2 - M \leq R(q^n - 1) < M^2 + M. \qquad (15)$$

**Example 4.2** *Consider the following binary code $C = \{000, 001, 010, 100\}$ where $q = 2, n = 3$. The mutiset $S_C$ of all different pairs of codeword differences of the code is*

$$S_C = \{001, 001, 010, 010, 100, 100, \\ 011, 011, 101, 101, 110, 110\}.$$

*Any nonzero element of $GF(2^3)$ appears at most two times in the multiset $S_C$, hence the code is 2-robust.*

*The code is not perfect since equality does not hold for (13). The code, however is an optimum $(3, 4, 2)_2$ Robust code. No other code can exist with the same $n$ and $R$ that has more codewords since 5 codewords would violate condition (13).*

## 5. ROBUST CODES WITH FLAT TOTAL AUTOCORRELATIONS

There is a strong relationship between robust codes, nonlinearity and nonlinear functions since all robust codes are nonlinear. We first review some basic definitions and properties of nonlinearity, a good survey of nonlinear functions can be found in [13].

Let $f$ be a function that maps elements from $GF(q^k)$ to $GF(q^r)$.

$$f : GF(q^k) \rightarrow GF(q^r) : a \rightarrow b = f(a). \qquad (16)$$

The nonlinearity of the function can be measured by using derivatives $D_a f(x) = f(x + a) - f(x)$. Let

$$P_f = \max_{0 \neq a \in GF(q^k)} \max_{b \in GF(q^r)} Pr(D_a f(x) = b), \qquad (17)$$

where $P_r(E)$ denotes the fraction of cases when E occurs. The smaller the value of $P_f$, the higher the corresponding nonlinearity of $f$. For linear functions $P_f = 1$.

**Definition 5.1** *A function $f: GF(q^k) \rightarrow GF(q^r)$ has **perfect nonlinearity** if $P_f = q^{-r}$.*

The parameters of systematic codes depend on nonlinearity of the encoding functions, as shown by the next theorem.

**Theorem 5.1** *Let $f$ be a function with nonlinearity $P_f$ that maps $GF(q^k)$ to $GF(q^r)$ where $k \geq r$, the set of vectors resulting from the concatenation of $x_1, x_2 : (x_1, x_2 = f(x_1))$ where $x_1 \in GF(q^k)$ and $x_2 \in GF(q^r)$ forms a $(k+r, q^k, P_f q^k)_q$ robust systematic codes.*

**Proof** The error $e = (e_1, e_2), (e_1 \in GF(q^k), e_2 \in GF(q^r))$ will be masked iff $f(x_1+e_1)-f(x_1) = e_2, x_1 \in GF(q^k)$, which is exactly when $D_{e_1} f(x_1) = e_2$. □

From Theorem 4.1, there are $q^{n-k}$ errors which will never be masked by $(n, q^k, R)_q$ systematic codes. Thereby a more strict bound can be derived for systematic codes. In this case we have

$$M^2 - M \leq R(q^n - q^{n-k}). \tag{18}$$

It is easy to verify that if the encoding function $f$ is a perfect nonlinear function, systematic codes constructed as in Theorem 5.1 satisfy the equality in (18). These codes have flat total autocorrelation functions and have the maximum M for a given n and R. They are also optimum with respect to Definition 4.3.

**Corollary 5.1** *A systematic robust code $C = \{(x_1, x_2 = f(x_1)) | x_1 \in GF(q^k), x_2 \in GF(q^r)\}$ is optimum if the encoding function $f$ is a perfect nonlinear function.*

**Proof** From Theorem 5.1 and Definition 5.1, if $f$ is a perfect nonlinear function, the resulting code is a $(k + r, q^k, q^{k-r})_q$ robust code. The optimality of the code can be verified by using Definition (4.3). □

**Remark 5.1** *The nonlinearity of the encoding function $f$ for systematic codes corresponds to the worst case error masking probability of the codes. We have:*

$$P_f = \max_{e=(e_1,e_2), e_1 \neq 0} Q(e) = \max_{e \in GF(2^{k+r})} Q(e).$$

*where $e_1 \in GF(q^k), e_2 \in GF(q^r)$.*

The following two constructions are examples of optimum robust codes based on perfect nonlinear functions.

**Construction 5.1** ***(Quadratic Systematic Code)*** *Let $x = (x_1, x_2, \cdots, x_{2s}, x_{2s+1})$, $x_i \in GF(q^r), s \geq 1$. A vector $x \in GF(q^{(2s+1)r})$ belongs to the code iff*

$$x_1 x_2 + x_3 x_4 + \cdots + x_{2s-1} x_{2s} = x_{2s+1} \tag{19}$$

*The resulting code is a $((2s+1)r, q^{2sr}, q^{(2s-1)r})_q$ optimum robust code.*

**Proof** The encoding function $f(x_1, x_2, \ldots, x_{2s}) = x_1 x_2 + x_3 x_4 + \cdots + x_{2s-1} x_{2s}$ is a perfect nonlinear function with $P_f = 1/q^r$ [13]. From Theorem 5.1 the resulting code is $R = q^{2sr}/q^r = q^{(2s-1)r}$ robust code. □

We will discuss applications of binary quadratic systematic codes for designing of memories with self-detection in Section 9.1.2, for data transmission in noisy channel in Section 9.2 and for data verification in Section 9.3.

**Example 5.1** ***(Robust Parity)****Methods based on linear parity check codes are often used for on-line error detection in combinational circuits [14]. The linear 1-parity codes can detect all errors of odd multiplicities but offer no protection for errors of even multiplicities. For devices and environments where the error distributions are unknown or non stationary the linear 1-parity codes can result in unpredictable behavior in the presence of errors.*

*As an alternative to the linear parity codes, the quadratic systematic robust codes defined in Construction 5.1 can be used. Taking $r = 1, q = 2$ the encoding function becomes bent function [13] and the resulting $(2s+1, 2^{2s}, 2^{2s-1})_2$ robust systematic code (robust parity code) has the same redundancy as the linear parity codes. Unlike their linear counterparts, robust parity codes can detect any error with a probability of at least $\frac{1}{2}$.*

**Construction 5.2** ***(Robust Duplication Code)*** *Let $x = (x_1, x_2)$ where $x_1, x_2 \in GF(q^r)$, $q$ is a power of an odd prime. The robust duplication code $C$ contains all vectors $x \in GF(q^{2r})$ which satisfy $x_1^2 = x_2$. The code is a $(2r, q^r, 1)_q$ optimum robust code.*

**Proof** The encoding function $f(x_1) = x_1^2$ is a perfect nonlinear function with $P_f = 1/q^r$ [13] for nonbinary fields. From Theorem 5.1 the resulting code is a $R = q^r/q^r = 1$ robust code. □

Perfect nonlinear functons, exist only for very limited parameters. For the case of binary codes, for example, there are no perfect nonlinear functions from $GF(2^k)$ to $GF(2^k)$ [13] [15]. Functions with optimum nonlinearity in this case are called almost perfect nonlinear (APN) functions [16], which have $P_f = 2^{-k+1}$. When $f$ are APN functions in the binary field, the robust codes construted as in Theorem 5.1 have $R = 2$. These codes are not optimum.

**Example 5.2** ***(Binary Robust Duplication Code)*** *In the binary field, $f(x_1) = x_1^2$ is not a perfect nonlinear function. Instead, we use $f(x_1) = x_1^3$ or $f(x_1) = x_1^{-1}$ as the encoding function which are almost perfect nonlinear. The parameter of the binary robust duplication code is $(2r, 2^r, 2)_2$.*

Robust duplication codes can be a viable alternative to standard duplication techniques. Application of binary robust duplication codes to memories with self-error-detection and comparison between standard and robust duplication techniques will be discussed in Section 9.1

## 6. CONSTRUCTION OF NEW CODES FROM OLD

We begin this section by introducing two modifications for systematic robust codes.

**Construction 6.1** ***(Augmented Code)*** *Let $C$ be a $(n, q^k, R)_q$ systematic robust code defined by the encoding function $f$. The augmented code $C^{(a)} = C \cup \{(\mathbf{0}, \beta) | \beta \neq f(\mathbf{0}), \beta \in GF(q^{n-k})\}$ where $\mathbf{0}$ is the all-zeros vector in $GF(q^k)$, is a $(n, q^k + q^{n-k} - 1, \max(q^{n-k}, R+2))_q$ robust code.*

**Proof** In the multiset $S_{\{(\mathbf{0},\beta \neq f(\mathbf{0}))\}}$ of vector differences of the set $\{(\mathbf{0}, \beta \neq f(\mathbf{0}))\}$ , each element is in the form $(\mathbf{0}, \beta), \beta \neq f(\mathbf{0}), \beta \in GF(q^{n-k})$ and appears exactly $q^{n-k} - 2$ times. From the Proof of Theorem 4.1 $(\mathbf{0}, \beta) \notin S_C$. Thereby the two multisets are disjoint. In the multiset of differences of vectors of $C$ and the additional vectors each element can appear at most two times. Hence, multiset of the augmented code $C^{(a)}$ contains each element in the form $(\mathbf{0}, \beta)$ exactly $q^{n-k}$ times and all other elements at most $R + 2$ times. □

**Remark 6.1** *Augmenting is only useful for codes with relatively large $R$. Augmented quadratic systematic codes have parameters $((2s+1)r, q^{2sr} + q^r - 1, q^{(2s-1)r} + 2)_q$. When $s = 1$, they are optimum.*

**Construction 6.2** ***(Punctured Code)*** *Let $C$ be a $(n, q^k, R)_q$ robust systematic code. Denote by $C_p^*$ the punctured code obtained by deleting $p < (n-k)$ check digits from each codeword of $C$. $C_p^*$ is a $(n-p, q^k, R_p^* \leq q^p R)_q$ robust code.*

To prove the result we start with a Lemma.

**Lemma 6.1** *Let $f : GF(q^k) \rightarrow GF(q^r)$ be a function with nonlinearity $P_f$. The punctured function $f^* : GF(q^k) \rightarrow GF(q^{r-1})$ formed by deleting one digit from the output of $f$ has a nonlinearity of $P_{f^*} \leq qP_f$. If $f$ is a perfect nonlinear function then so is $f^*$.*

**Proof** The nonlinearity of the function $f$ is defined as $P_f = \max_{0 \neq a \in GF(q^k)} \max_{b \in GF(q^r)} Pr(D_a f(x) = b)$ where $D_a f(x) = f(x+a) - f(x)$. Deleting one digit from the output of the function $f$ will cause inputs which were previously mapped to outputs differing in the deleted digit to be remapped to the same output. Hence the derivitives of these elements which only differed in the deleted digit will be equal. The nonlinearity of $f^*$ is therefore

$$P_{f^*} = \max_{0 \neq a \in GF(q^k)} \max_{i \in GF(q^{r-1})} \sum_{j=0}^{q-1} Pr(D_a f(x) = (i, j)), \tag{20}$$

where $(i, j)$ is the concanetantion of $i \in GF(q^{r-1})$ and $j \in GF(q)$ . Since $Pr(D_a f(x) = (i, j)) \leq P_f$ for any $a$ and $(i, j)$ we have $P_{f^*} \leq qP_f$.

When $f : GF(q^k) \rightarrow GF(q^r)$ is a perfect nonlinear function, $P_f = q^{-r}$ and $P_{f^*} \leq q^{-r+1}$. Since for a functions which maps $GF(q^k) \rightarrow GF(q^{r-1})$, the nonlinearity is lowerbounded by $q^{-r+1}$, $P_{f^*} = q^{-r+1}$ and $f^*$ is perfect nonlinear. □

The proof of Construction 6.2 easily follows from the above Lemma.

**Proof** From Lemma 6.1 deleting $p$ bits from the range of the encoding function $f$ results in a function with $P_{f_p^*} \leq q^p P_f$. From Theorem 5.1 the relationship between a systematic $(n, M, R)_q$ robust code and the nonlinearity of its encoding functions is $P_f = R/M$. □

**Corollary 6.1** *Punctured robust codes formed by deleting $p < n - k$ check symbols from optimum systematic robust codes based on perfect nonlinear functions are also optimum.*

Punctured robust codes are still systematic codes. They are optimum and have flat total autocorrelations as long as the encoding function of the original code is a perfect nonlinear function. Augmented robust codes, on the other hand, are not systematic codes. It is relatively difficult to implement encoding and decoding for these codes. Augmented quadratic systematic code is an example of optimum nonsystematic robust codes. Parameters of some optimum punctured and augmented robust codes are given in Table 3.

We next show two modification methods based on the properties of autocorrelation functions that can generate new perfect robust codes from old ones.

Let $f(x) : GF(q^n) \rightarrow \{0, 1\}$ be a boolean valued function and $\sum_{x=0}^{q^n-1} f(x) = M$. Let $\overline{f} = 1 + f \pmod 2$ be the inversion of $f$. Then

$$B_{\overline{f}}(e) = q^n - 2M + B_f(e). \tag{21}$$

If $f$ is the characteristic function $\chi_c(x)$ of a code $C \in GF(q^n)$, $\overline{f}$ will be the characteristic function $\overline{\chi}_c(x)$ of $\overline{C}$, which is the complement code of $C$ containing all vectors in $GF(q^n)$ that do not belong to $C$. Thus we have:

**Corollary 6.2** *Let $C$ be a $(n, M, R)_q$ robust code. Then its complement code $\overline{C}$ is a $(n, q^n - M, q^n - 2M + R)_q$ robust code. Moreover, $\overline{C}$ is perfect if an only if $C$ is perfect.*

**Proof** Recall that the robustness of the code $C$ is

$$R = max_{e \in GF(q^n), e \neq 0} B_{\chi_c}(e), \tag{22}$$

where $\chi_c$ is the characteristic function of the code. Denote by $\overline{R}$ the robustness of $\overline{C}$. From theorem 6 we have:

$$\begin{aligned} \overline{R} &= \max_{e \in GF(q^n), e \neq 0} B_{\overline{\chi}_c}(e) \\ &= \max_{e \in GF(q^n), e \neq 0} (q^n - 2M + B_{\chi_c}(e)) \\ &= q^n - 2M + \max_{e \in GF(q^n), e \neq 0} B_{\chi_c}(e) \\ &= q^n - 2M + R \end{aligned}$$

Obviously, $\overline{C}$ will have "flat" autocorrelation if and only if $C$ does. □

We next give an example of modifications which do not generate codes with new parameters but only change the structure of the code. This method is due to the fact that autocorrelation functions are invariant to the affine transformation of variables.

**Theorem 6.1** *[6] Let $f : GF(q^k) \to GF(q^r)$ and $\sigma = (\sigma_{ij})$ a q-ary matrix ($i, j = 0, 1, \cdots, k-1$), $|\sigma|_q \neq 0$ ($|\sigma|_q$ denotes the determinant of $\sigma$ over $GF(q)$). $\tau$ is a q-ary vector of length k. Denote by $\phi(x) = f(\sigma \otimes x + \tau)$, where $\otimes$ is the matrix multiplication and "+" is the componentwise addition over $GF(q)$. Then*

$$B_\phi(x) = B_f(\sigma \otimes x). \tag{23}$$

If $f = \chi_c(x)$ is the characteristic funciton of a code $C \in GF(q^n)$ with codewords $x = (x_1, x_2, \cdots, x_n)$, the by the above theorem the affine transformations of $(x_1, x_2, \cdots, x_n)$ do not change the "value distribution" of the autocorrelation and thus do not change $R$. Thereby we have

**Corollary 6.3** *Let $x = (x_1, x_2, \cdots, x_n)$ be the codewords of a $(n, M, R)_q$ robust code, $\sigma$ be a $n \times n$ nonsingular matrix and $\tau$ be a vector of length k over $GF(q)$. The new code constructed by applying an affine transformation $\sigma \otimes x + \tau$ to x is still a $(n, M, R)_q$ robust code.*

Although not able to generate codes with new parameters, the above modification method enbles us to find codes that are easier to implement in hardware.

**Example 6.1** *Consider a $(5, 2^4, 2^3)_2$ optimum systematic robust codes with codewords $x = (x_1, x_2, x_3, x_4, x_5), x_i \in GF(2)$ defined by the following encoding function*

$$x_1x_2 + x_2x_3 + x_1x_3 + x_3x_4 = x_5. \tag{24}$$

*To implement this code in hardware, we need 4 2-input AND gates and 3 2-input XOR gates for the encoder. Let $\sigma$ be*

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

*After applying the linear transformation $\sigma \otimes x$ to x, the encoding function becomes $x_1x_2 + x_3x_4 = x_5, x_i \in GF(2)$, which is more organized and easier to implement. Compared with the original code, only two 2-input AND gates and one 2-input XOR gates are needed for the encoder, which cuts down the hardware redundancy by more than 50 %. The new code is a quadratic systematic code and still has the parameter $(5, 2^4, 2^3)_2$.*

## 7. CONSTRUCTION OF BINARY ROBUST CODES WITH $R = 2$

Robust codes with $R = 2$ in $GF(2^n)$ have the minimum possible $R$ in binary field. In previous sections, we have presented several construction methods which can generate $(n, M, 2)_2$ robust codes, ie. robust duplication codes when $q = 2$. In this section, we will show that $(n, M, 2)_2$ robust codes can be constructed from parity check matrices of linear codes with hamming distance $d = 5$. We refer to linear codes by a triple $[n, M, d]_q$ where n is the code's dimension, $M$ the number of codewords and $d$ the minimum distance.

**Theorem 7.1** *The columns of a parity check matrix H of a $[M, 2^k, 5]_2$ linear code form a $(M - k, M, 2)_2$ robust code. Moreover, the all-zeros vector in $GF(2^{M-k})$ can be further added to construct a $(M-k, M+1, 2)_2$ robust code.*

**Proof** Every four columns of a parity check matrix $H$ of a $[M, 2^k, 5]_2$ linear code are linearly independent. In binary field it means

$$h_a + h_b + h_c + h_d \neq 0, \tag{25}$$

or

$$h_a + h_b \neq h_c + h_d. \tag{26}$$

where $a, b, c, d \in [0, M-1)$ are different integers, $h_i$ are column vectors of $H$. Recall that for a $[M, 2^k, d]_2$ linear code, $H$ is a $(M-k) \times M$ matrix and each $h_i$ is a $M-k$ binary vector. If we use $h_i$ as the codeword of a robust code, from (26) it is clear that in the multiset $S$ of the robust code, every nonzero element in $GF(2^{M-k})$ appears no more than twice. Hence the robust code composed of all column vectors of H has the parameters $(M - k, M, 2)_2$.

It is also true that for $[M, 2^k, 5]_2$ linear codes,

$$h_a + h_b \neq h_c, \tag{27}$$

which means no $h_i$ belongs to the multiset S of the $(M - k, M, 2)_2$ robust code. After adding the all-zeros vector in $GF(2^{M-k})$, every $h_i, i \in [0, M-1)$ will appear twice in $S$. $R$ of the robust code will not increase. Thereby the resulting code is a $(M - k, M + 1, 2)_2$ robust code. □

**Remark 7.1** *The above theorem is also true for nonbinary case.*

**Theorem 7.2** *$[M, 2^k, d \geq 5]_2$ linear codes can be constructed from $(M - k, M + 1, 2)_2$ robust codes.*

**Proof** Without lost of generality, we assume that the robust code contains the all-zeros vector in $GF(2^{M-k})$. (Otherwise the code can be shifted such that it will contain the all-zeros vector.) The parity check matrix H of the linear code can be constructed using all nonzero codewords of the robust code as its column vectors. The dimension of H is $M - k$ because all codeword components of robust codes are linear independent. Follow the same analysis as in the proof of Theorem 7.1, it is easy to verify that the resulting linear code has length $M$ and hamming distance at least 5. It is a $[M, 2^k, d \geq 5]_2$ code. □

We notice that the problem of finding the largest size of robust codes with $R = 2$ is strongly connected to the well-known open problem in coding theory, which is

- Given $k$ and $d = 5$, find the largest possible $M$ for which a $[M, k, 5]_2$ linear code exists.

Another problem that is closely related to the above one can be:

- Given $M$ and $d = 5$, find the largest possible $k$ for which a $[M, k, 5]_2$ linear code exists.

For binary case, some of the known longest linear codes with $d = 5$ are presented in [17]. The values of the largest possible $k$ for $M \leq 33, d = 5$ are shown in [18].

## 8. PARTIALLY ROBUST CODES

Robust codes have higher complexity of encoding and decoding than classical linear codes. The $(n = (2s + 1)r, 2^{2sr}, 2^{(2s-1)r})_2$ quadratic systematic codes in binary field from Construction 5.1 require $s$ $r$-bit multipliers and $s-1$ $r$-bit componentwise additions. Assuming a $r$-bit multiplier requires $r^2$ two-input gates the encoder for these systematic quadratic code can be implemented with $sr^2 + r(s-1)$ 2-input gates.

As a tradeoff between robustness and the hardware overhead for computational devices, *partially robust codes* were introduced in [3]. These codes combine linear and nonlinear mappings to decrease the hardware overhead associated with generation of check bits. The encoding of systematic partially robust code is performed first by using a linear function to compute the redundant check $r$ check symbols followed by nonlinear transformation. The use of the linear code as the first step in the encoding process typically results in hardware savings in the encoder or predictor since the nonlinear function needs to only be computed based on the output of the linear block, which has length $r$ that is much shorter than $n$. The application of the nonlinear transformation reduces the number of undetectable errors thus increasing the robustness of the linear codes.

**Construction 8.1** *(Partially Robust Codes) Let $f : GF(q^r) \rightarrow GF(q^r)$ be a nonlinear function with $P_f < 1$ and let $H : GF(q^k) \rightarrow GF(q^r)$ , $r \leq k$ be a linear onto function. All vectors in the form $(x, f(H(x)))$ form a partially robust code in $GF(q^{k+r})$. The set of undetectable errors is a $q^{k-r}$ subspace of $GF(q^{k+r})$, $q^k - q^{k-r}$ errors are detected with probability 1, and remaining $q^{k+r} - q^k$ errors are detected with probability at least $1 - P_f$.*

**Proof** Error $e = (e_1, e_2), e_1 \in GF(q^k), e_2 \in GF(q^r)$ is masked if and only if

$$f(H(x + e_1)) = f(H(x)) + e_2, \tag{28}$$

or

$$f(H(x) + H(e_1)) = f(H(x)) + e_2. \tag{29}$$

If $H(e_1) = e_2 = 0$, (29) is always satisfied and the error will be masked with probability 1. Since $H$ is a linear onto function, the number of errors $e = (e_1, e_2)$ (including 0) satisfying $H(e_1) = e_2 = 0$ is $q^{k-r}$ and these errors form a $k - r$ dimensional subspace of $GF(q^{k+r})$.

If $H(e_1) = 0$, $e_2 \neq 0$, then (29) will never be satisfied. These errors will be detected with probability 1. The number of these errors is $q^k - q^{k-r}$.

If $H(e_1) \neq 0$, from (17), we know that given $e = (e_1, e_2)$ there are at most $P_f q^r$ values of $H(x)$ satisfying (29). Because $H$ is a linear onto function, for each value of $H(x)$, there are $q^{k-r}$ possible solutions for $x$. Thereby errors in this class will be masked no more than $P_f q^r \cdot q^{k-r} = P_f q^k$ times. They will be detected with probability at least $1 - \frac{P_f q^k}{q^k} = 1 - P_f$. $\square$

The number of undetectable errors is reduced from $q^k$ to $q^{k-r}$ compared to the linear code with the same redundancy. The combination of a linear functions simplifies the prediction complexity for devices with linear or partially linear functions.

Application of partially robust codes for error-detection in memories will be discussed in Section 9.1.2. Partially robust codes with $k = 128$ and $r = 32$ have been used in [4] for design of private key security devices based on Advanced Encryption Standard (AES) resistent to fault injection attacks. Implementation of this approach resulted in about 80% hardware overhead.

## 9. ERROR DETECTION AND DATA COMPRESSION BY ROBUST CODES

Robust error detecting codes have, by design, a uniform or almost uniform error-detection coverage and data dependent error detection capability. These unique properties make the codes useful for channels with unknown or non-stationary error distributions, and channels with highly correlated or "lazy" errors. To illustrate the potential benefits we provide analysis for applications where the codes show a benefit over classical linear codes. The codes in this section are all binary codes.

### 9.1. Robust Error Detection in Memories with Unknown or Non-Stationary Error Distributions

The error characteristics in silicon devices are changing and in many instances can be unpredictable. Using the case of memories as an example of a channel with unknown or non-stationary error distributions we demonstrate the possible benefits and methods of adding robust codes to the designs.

As devices are being pushed into the deep submicron technologies reliability is increasingly becoming a critical design issue. Aggressive technology scaling is causing transient effects to have a larger impact on the overall reliability of a circuit. Cross coupling, ground bounce, and external radiation are creating more and more unpredictable transient and soft errors [19]. Memory devices such as DRAMs and SRAMs are especially vulnerable. Decreased voltage levels and increased densities mean that there is a higher probability of a transient pulse

to get latched and become a permanent bit flip in a memory cell. With predicted densities of 64GB memories per chip by the year 2008, memories are expected to face increased reliability challenges [20].

Furthermore, as embedded systems are becoming ubiquitous, their roles are also becoming more mission critical for military, automotive, aerospace, and medical applications. Many embedded memories are exposed to more strenuous and unpredictable environments while their reliability becomes a matter of critical importance and safety. Many medical and mobile devices, for example, can be subjected to various environments in short spans of time. In a matter of minutes a memory device can be moved from sea level, where single event rates from cosmic rays are low, to being on a trans-Pacific flight where the error rate can increase 300x [19].

Despite the fact that error characteristics of memories have been less and less predictable not much has been done to protect these memories in the face of such errors. Most of the memory protection schemes today are designed for a given error model or error distribution. These systems usually assume single or double errors and use simple codes (such as duplication or extended Hamming codes) which concentrate their error detecting power for those errors. These protection methods offer a good protection for errors of low multiplicities but have a poor performance for other error distributions.

In many situations multiple errors in a word can occur. Things such as address decoder errors, power glitches, and cross-couplings can create difficult to predict multi-bit errors. Different protection methods are necessary when the error distributions are unknown or non-stationary as is the case for embedded memories which are exposed to a wide range of changing environments.

In the next section we analyze the limitations of existing error detection techniques and show how robust error-detecting codes can be applied to memory architectures to make memories more robust and their error detection more predictable in the presence of unpredictable environments where the error distributions are unknown or non-stationary.

### *9.1.1. Self Error Detection in Memories by Linear Error Detection Codes*

Two commonly used error detection schemes used in memories today are duplication and the extended Hamming codes [19]. Both of these schemes are based on very simple linear codes to reduce the hardware and time overheads.

The high level architecture for protected memory is shown in Figure 3. Extra redundant memory blocks are added to store redundant encoded data (the signature of the data). In the case of duplication ($k = r$), no encoder is necessary since the same data is stored in both copies. For other linear systematic codes the encoder performs matrix multiplication over $GF(2)$ between the k-bit data and the parity check matrix of the selected linear code. The same encoder can be used to recreate the signature of the data for error detection in the Error Detecting Network (EDN).

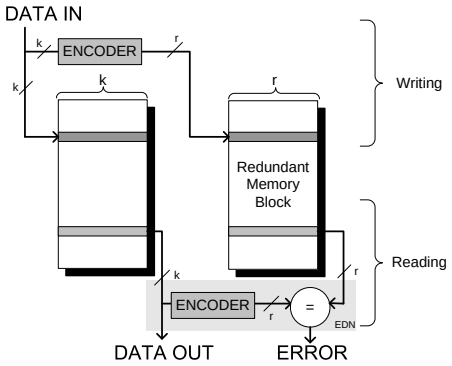



Figure 3. High level memory architecture with self error detection

When the linear codes are used in the protection of memories it is typically assumed that only single or double bit errors in a word will be observed. For these types of errors the current methods can provide for good protection. However, when the error distributions change and when errors of higher multiplicities occur the methods can have unpredictable behaviors.

As an example, Figure 4 shows the percent of detectable errors as a function of error multiplicity (number of distorted bits) for 8-bit duplication ($k = r = 8$) and the the linear $[72, 2^{64}, 4]_2$ extended Hamming code. The detection capability of both codes depends largely on the multiplicity and type of the error. The schemes offers relatively poor protection for errors of even multiplicities. Other error distributions can result in more uneven or unpredictable error detection profiles. For linear codes, for example, any error which is a codeword will always be masked. For errors of this type the error detection based on linear codes will provide no protection.

### *9.1.2. Self-Error Detection in Memories by Robust Codes*

Figure 5 shows the probability of detecting an error as a function of error multiplicity (number of distorted bits) for codes based on the robust codes presented in the previous section. While these codes have the same parameters ($k$ and $r$) as the linear codes, the error detection profile is much more uniform providing a more predictable error detection capability regardless of the error distributions.

The error detection profiles of Figure 5 were generated by simulating the codes with random messages and random errors of given multiplicities. The simulations for the 8-bit duplication code were performed exhaustively for all possible message/error pairs. The other two graphs show the result of simulating 200,000 error/message pairs for each error multiplicity.

Figure 5a shows the error detection profile for a $(16, 2^8, 2)_2$ robust duplication code. This code is robust and any error is detectable. Compared to the linear duplication code with the same $n$ and $k$ the code has almost completely uniform error detection. This robust code has R=2, meaning that any error can be masked for at most two messages. Unlike for the linear codes, regardless of what

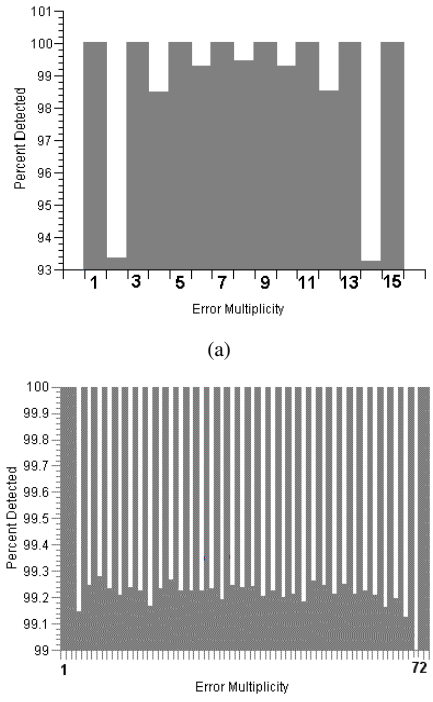



Figure 4. Percentage of errors detected versus error multiplicity for (a) 8-bit linear duplication (b) $[72, 2^{64}, 4]_2$ extended Hamming code

subset of errors is chosen for this robust code the error masking probability is bounded by $2^{-7}$.

Figure 5b shows the error detection profile for a $(72, 2^{64}, 2^{56})_2$ robust quadratic code which has the same $n$ and $k$ as the linear extended Hamming code. Unlike the linear Hamming code which has a total of $2^{64}$ undetectable errors and whose error detection depends heavily on the multiplicities of errors the robust code is capable of detecting all errors and has an almost completely uniform error detection.

Finally, Figure 5c presents the error detection profile for a partially robust code where the linear onto function is the parity check matrix of the extended Hamming code. Compared to the original linear extended Hamming code, the partially robust code has a much flatter error detection profile, but not as flat as the robust code ( see Figure 5b).

Table 1 summarizes and compares the error masking probabilities of the proposed robust error detection method and the traditional methods based on linear error detecting codes. The table compares the error masking probabilities between the $[128, 2^{64}, 2]_2$ linear duplication code and the corresponding $(128, 2^{64}, 2)_2$ robust code and between the linear $[72, 2^{64}, 4]_2$ Hamming code and the corresponding partially robust extended Hamming code when all the codewords are assumed equiprobable.

As the table shows for the robust duplication code there are no undetectable errors. For the partially robust code based on the Hamming code the number of undetectable errors is drastically reduced.

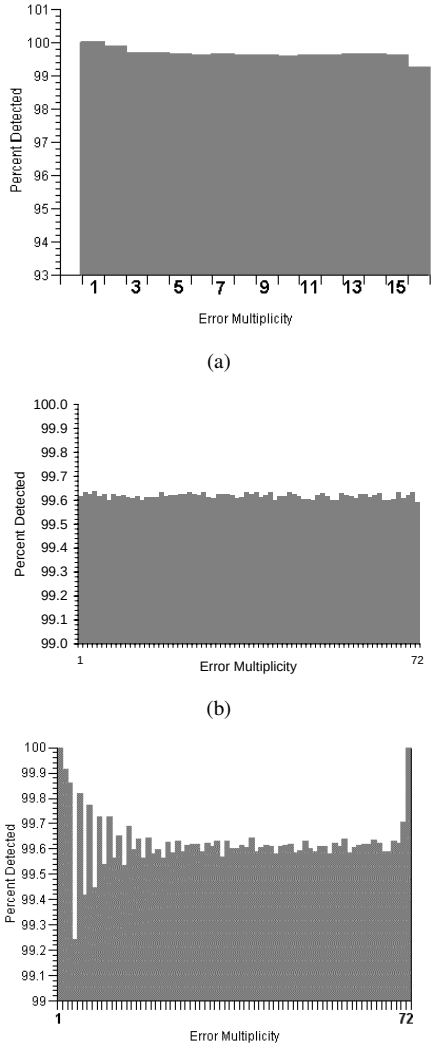



Figure 5. Percentage of errors detected for (a) $(16, 2^8, 2)_2$robust duplication code (b) $(72, 2^{64}, 2^{56})_2$ robust quadratic code (c) partially robust $(72, 2^{64})_2$ extended Hamming code

### 9.2. Robust Error Detection in Lazy Channels with Memory

In addition to providing a more uniform error detection coverage compared with classical linear codes robust error detecting codes have better detection characteristics in channels where errors have a high laziness or probability of repeating themselves.

**Definition 9.1** ***(Laziness)*** *The* ***laziness*** $L(e)$ *of an error* $e$ *in a channel is the conditional probability that if an erroneous output* $\tilde{x}(i)$ *was a result of an error* $e$*, the next erroneous output* $\tilde{x}(i+1)$ *was also the result of the same error*

$$L(e) = Pr(e = \tilde{x}(i+1) - x(i+1) | e = \tilde{x}(i) - x(i)), \quad (30)$$

*where* $x(i) \neq x(i+1), e \neq 0$ *and all messages* $x$ *are considered equiprobable.*

Table 1. Probability of Masking for Linear and Robust Codes of Length 128 and 72

| | $\max Q(e)$ | number of undetectable errors |
|---|---|---|
| $[128, 2^{64}, 2]_2$ linear duplication | 1 | $2^{64}$ |
| $(128, 2^{64}, 2)_2$ robust duplication | $2^{-63}$ | 0 |
| $[72, 2^{64}, 4]_2$ linear extended Hamming | 1 | $2^{64}$ |
| $(72, 2^{64})_2$ partially robust Hamming | 1 | $2^{56}$ |
| $(72, 2^{64}, 2^{56})_2$ robust quadratic | $2^{-8}$ | 0 |

The above definition of laziness can be extended to include non stationary probabilities and dependence of laziness on messages and other effects, but this simple definition of laziness allows a cleaner demonstration of benefits of robust codes.

Errors with a high laziness can occur in many hardware implementations. Faults in linear networks consisting of XOR gates only or fanout-free logic implementations will often result in internal faults manifesting themselves as repeating errors at the outputs of the devices. Failures in interconnect networks such as busses can also result in repeating errors since faults can directly manifest themselves as errors. For such devices a single fault has a very high probability of manifesting itself in a constant error pattern regardless of the data it distorts.

Errors with high laziness can also occur in channels where an adversary is the cause of the malfunctions. It has been shown that security of implementations of cryptographic algorithms can be compromised if faults occur in the devices [21]. Malicious attackers can inject faults to help in cryptanalysis. Due to the mechanical nature of the fault injection the fault injection will typically be much slower than the operation of the device often causing a single fault or error to affect several cycles of data processing. This inherent slowness or laziness of fault injection has been used as one of the motivations for the use of robust codes in cryptographic hardware [4].

**Example 9.1** ***(Circuit errors in combinational networks resulting from single stuck-at zero faults)***

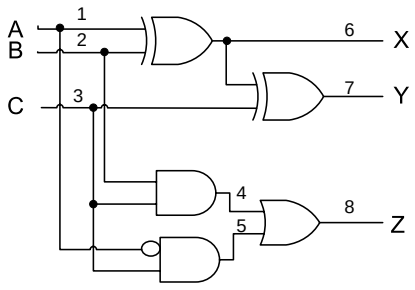



Figure 6. A circuit with laziness

*Consider the circuit in Figure 6. In the presence of single stuck-at-zero faults the errors at the output of the circuit can exhibit high laziness and the errors are not limited to single bit distortions. The analysis of the laziness of all the errors in the presence of a single stuck-at-zero fault at each of the possible eight locations is shown in Table 2. The first eight srows in Table 2 shows the $L(e)$ for each error when a single stuck-at-zero fault affects the numbered location in the circuit. For each fault location some errors exhibit very high laziness while other errors are do not apear for the specific fault (represented by a dash in the table). The final row shows the laziness of each error if any of the single stuck-at-zero faults and all input vectors are equiprobable. When any of the single stuck-at-zero faults can occur, for example, when the error of 110 is observed at the output, there is a 55.5% probability that the next erronous output will be distorted by the same error.*

The repeating nature of the errors can be problematic for classical error detection methods since when the manifestation of faults is not detectable by the code for one output vector then the corresponding error will never be detected and the device will function erroneously without providing any detection of a possible malfunction.

For a robust code all errors are detectable. For a binary systematic $(n, 2^k, R)_2$ robust code any error is masked with a probability of no worse than $R/2^r$ when the messages are randomly chosen and uniformly distributed. Moreover, since the detection of each error is message dependent, the error masking probability decreases the more messages the error affects. Figures 7 show the effect of error detection when an error is stationary for multiple messages. The probability of detection increases exponentially with the the number, $t$, of different data being affected by the same error. For robust duplication, as is shown in Figure 7a, which results in a 2-robust error-detecting code all possible errors are detectable if they remain constant for $t \geq 3$ messages. The error detection profile for $t = 2$ messages for the $(72, 2^{64}, 2^{56})_2$ robust code is shown in Figure 7b. After two messages the probability of masking an error is reduced to $2^{-16}$ for any error.

Likewise, Figure 7c shows the probability of detecting an error for the partially robust $(72, 2^{64})_2$ extended Hamming code when an error is present for $t = 2$ different messages. While the application of the non-linear transformation on the redundant bits of the extended Hamming code does not result in a completely robust code, the number of undetectable errors is reduced from $2^k$ to $2^{k-r}$ and the probability of error masking is reduced when the error

Table 2. Laziness of errors $L(e)$ resulting from single stuck-at-zero fault for circuit from Figure 6

| location of fault | e=001 | e=010 | e=011 | e=100 | e=101 | e=110 | e=111 |
|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | 0.333 | 0.333 |
| 2 | - | - | - | - | - | 0.333 | 0.333 |
| 3 | - | 0.333 | 0.333 | - | - | - | - |
| 4 | 1 | - | - | - | - | - | - |
| 5 | 1 | - | - | - | - | - | - |
| 6 | - | - | - | - | - | 1 | - |
| 7 | - | 1 | - | - | - | - | - |
| 8 | 1 | - | - | - | - | - | - |
| total laziness $L(e)$ | 1 | 0.666 | 0.333 | - | - | 0.555 | 0.333 |

is present for multiple messages, a property not found in the linear Hamming code.

Robust codes can offer an advantage over the classical linear error detection in the latency of fault detection providing the system with and earlier notification of malfunction. The detection properties of linear and robust codes are compared for two metrics in the next two subsections.

#### *9.2.1. Fault Detection Probability for Linear and Robust Codes*

For many reliable systems it is critical to detect as many occurrences of a failure as possible. Multiple instances of the same error for multiple consecutive messages are likely to occur due to a single fault in many devices. The larger the laziness $L(e)$ of errors in a channel the longer is the average span of the errors or rather the more likely that the error will affect more than one message. Detection of at least one instance of the error in its span allows for a detection of a failure.

For a classical linear error detecting code the laziness of a channel does not improve the error detection probability of the code. That is, if the error is masked for the first message the error will be masked for the rest of the messages within the span of the error. Hence if the probability of masking an error for one message is $Q(e)$, the probability of masking the whole span is $Q(e)$ regardless of the laziness of the error for the linear code. If a failure results in an undetectable error with a high laziness, the error can affect many messages and the system will have no indication that a failure occurred. Classical linear error-detecting codes do not take an advantage of repeated manifestations of an error.

For a robust code the probability of detecting an error increases as the error affects more messages or rather as the span of the error increases. Since the error detection of error is data dependent, the more messages the error affects the higher the chances of detecting the error at least once within the span.

If the probability of masking an error $e$ for a a robust code for a single message is $Q(e)$ then the probability that an error will be masked for its entire span is

$$Q(e, L(e)) = Q(e)(1 - L(e)) \sum_{i=0}^{\infty} L(e)^i Q(e)^i, \quad (31)$$

assuming as before that messages are equiprobable, the geometric series can be simplified to

$$Q(e, L(e)) = Q(e)(1 - L(e)) + \frac{Q(e)^2(1 - L(e))L(e)}{1 - Q(e)L(e)}. \quad (32)$$

For the robust codes as the $L(e)$ increases the probability of detecting the error at least once in the span of the errors increases as well, providing for a higher likelihood of detecting failures that can manifest themselves as repeating errors.

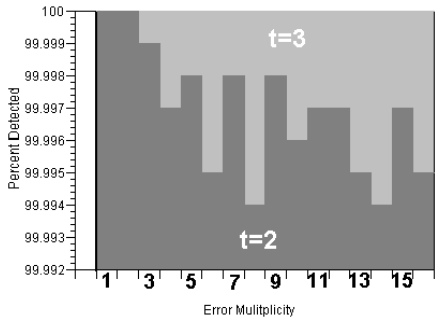
#### *9.2.2. Fault Detection Latency for Linear and Robust Codes*

Minimizing latency of failure detection, the time between when a failure manifest itself as a nonzero error and the when it is detected, is another important consideration for on-line detection.

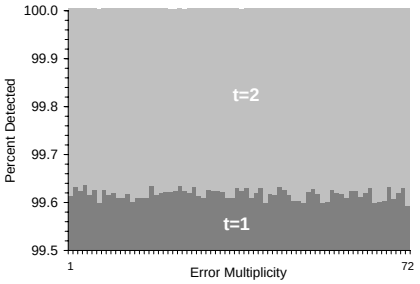
When a failure within a device results in erroneous outputs it is important to reduce the number of erroneous messages which will be processed until the failure is detected by detecting an error. The detection latency for robust codes is not affected by the laziness $L(e)$ of the manifested errors. The latency of detection for classical linear codes, however, increases as the laziness of errors increases.

To quantify this metric we analyze linear and robust codes with respect to the probability of detecting at least one error in a span of erroneous messages assuming an error laziness $L(e)$ in the channel.

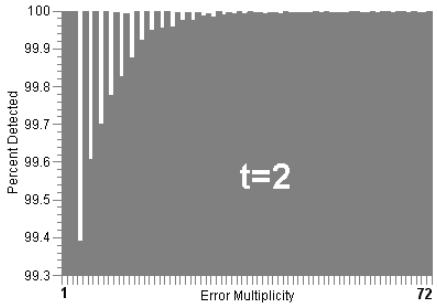
For a linear code the probability of detecting at least one error in a span of $T$ message/error pairs is dependent on the number of different errors present within the span. If in the span of $T$ error/message pairs only $t$ errors $e_1, \cdots, e_t$ were unique the probability of masking all errors within the span is $\prod_{i=1}^{t} Q(e_i)$. The larger the laziness of errors in a channel the fewer distinct errors will be




(a)




(b)




(c)

Figure 7. Percentage of errors detected for (a) 8-bit robust duplication after t=2 and 3 messages (b) $(72, 2^{64}, 2^{56})_2$ robust code (c) partially Robust Hamming code after t=2 data

observed and the error detection latency of the linear code will degrade as laziness of errors increases.

For a robust code on the other hand, the detection latency is not affected by the laziness of the errors. For a robust code it is the number of different message/error pairs which are observed and not the number of different errors which affect the latency of detection. In this case the probability of masking all errors for $T$ message/error pairs is $\prod_{i=1}^{T} Q(e_i)$ if all messages within the span are unique.

As the laziness increases the robust codes detection latency remains constant while there is a increase for linear codes.

### 9.3. Remote Data Verification by Robust Codes

Robust codes can be used to provide a guaranteed level of protection against an unbounded error model in data verification applications. Large sets of data can be reliably compressed and compared at remote locations using robust codes providing a provable guarantee of detection of set differences while minimizing communication complexity when no limits on errors are assumed.

To illustrate the data verification application consider a system designed for high reliability in which remote machines perform identical computations on identical data sets. The computations of one machine are mirrored by another remote and possibly different machine. To ensure reliability the remote machines can exchange intermediate results of computations at specific checkpoints in the program which is executed by both machines to ensure that no errors occurred. To make the checkpoint verification efficient the large intermediate data sets (e.g. contents of cache memories) cannot be directly exchanged but require compression before verification.

For the data used in computations even a single error or fault may result in large number of errors in the final results. In most cases, the exact error characteristics of the intermediate computations are are very difficult to predict or estimate even in the presence of single transient faults as errors can accumulate after several computations.

Robust codes can be used in such a situation to compress and verify remote data with a minimum probability of error masking regardless of the number of differences of the results of the intermediate calculations at the two remote machines. The encoding function of a systematic robust code can be used to compress the $k$ bits into a smaller $r$ bit signature ($r << k$) that can be used to verify the equality of the remote $k$-bit data to that of the local copy. The verification based on the compression using robust codes can provide for more uniform verification characteristics than a methods using compression based on linear error detecting codes which have been used in [22].

Data compression for verification using robust codes can be performed as follows. Assume machine A want to verify a checkpoint of large fixed length dataset X of $k$ bits with a remote machine B. To verify the consistency of the two data machine B compresses its dataset into $r$ ($r << k$) bits by generating a $r$-bit signature in such a way that the $k$ bits of the original data and the $r$-bit signature on machine B is a codeword of systematic $(k + r, 2^k, R)_2$ robust code. Machine B then sends only the data $r$-bit signature to the machine A. Machine A can then generate a $r$-bit signature for its own data set and compare the signature to the one received from machine B. We will call this approach robust check-point data verification.

**Theorem 9.1** *For robust verification of a $k$ bit data set based on a systematic $(n, 2^k, R)_2$ robust code (if only one of the machines has an error in its data set) the probability that the error will be missed during verification is at most $R/2^k$.*

**Proof** Distortion $e$ that affects only one of the machines will be masked if for the compression function $f$ and dataset

$X$ the following relation is satisfied $f(X+e) = f(X)$. From Theorem 5.1 any $(n, 2^k, R)_2$ systematic Robust code can be defined by the nonlinearity of its encoding function. For a systematic $(n, 2^k, R)_2$ Robust code the nonlinearity of the encoding function is $P_f = R/2^k$ where $P_f = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^r)} Pr(f(x+a) - f(x) = b)$.

We note that for any verification method based on linear compression the are $2^k$ undetectable distortions where the probability of masking is one.

In the case when both data sets A and B can both be distorted there are errors which cannot be detected. If both of the machines have an error in their computations then there are $2^k - 1$ possible errors which are undetectable. For any verification method which uses linear compression there are $2^k 2^{k-r}$ such errors.

**Example 9.2** *Consider the case of two remote machines computing on a data set of $k = 2^{24}$ bits. To verify consistency between the two data sets one machine computes the $r$-bit signature of the $k$ bits based on the encoding function of a systematic robust code. Taking the quadratic systematic robust code ( Construction 5.1 ) as an example, one machine would compute the quadratic signature which would correspond to the robust code with the desired detection parameters and redundancy. Using encoding based on a $(2^{24} + 64, 2^{2^{24}}, 2^{2^{24}-64})_2$ robust code the $k = 2^{24}$ data bits are compressed into $r = 64$ bits for transmission and the probability of masking any error is at most $2^{-64}$. The computation of the signature requires one $64$-bit multiplier and one $64$-bit adder/accumulator in $GF(2^{64})$ which should be used for $2^{17}$ clock to compute the signature.*

## 10. CONCLUSIONS

Functions with flat autocorrelation can be used to construct optimum codes for channels with unknown or nonstationary error distributions. The presented robust codes provide for uniform or almost uniform protection against all error patterns. Using properties of autocorrelation functions and perfect nonlinear functions the parameters of robust codes were analyzed and several constructions of the codes were presented.

The presented robust codes can be used for error detection and data compression. Increased feature size scaling of integrated circuits and the ubiquitious use of mobile computers makes the codes practical and useful for error detection in many hardware devices. We demonstrated the potential benefits of the codes over classical linear codes in memory applications, "lazy" channels, and robust check-point verfications.

Table 3 is a short summary of the construction methods and parameters of known robust codes. Clearly, for every $n$ and $q$ there is at least one nontrivial $R = R(n, q)$ such that an optimum $(n, M, R)_q$ robust code exists. From the practical point of view, the most important case is $q = 2$. When $n = 6, q = 2$, for example, $(6, 16, 4)_2$ and $(6, 19, 6)_2$ optimum robust codes can be constructed using Constructions 5.1 and 6.1. However, it is true that for some $n, q$ and $R$, optimum $(n, M, R)_q$ robust codes do not exist. The following problem is of great interest and is still open.

- For every $n$ and $q$, find all possible values of $R(n, q)$ such that an optimum $(n, M, R)_q$ robust code exist.

We also note that optimum systematic robust codes, which are more practical for applications than nonsystematic codes, exist for all even $k$ and arbitrary $q$. As the parameters of systematic robust codes are determined by the encoding function, the problem of constructing good systematic robust codes is strongly related to the problem of finding highly nonlinear functions.

## 11. ACKNOWLEDGMENTS

The authors would like to thank Dr. Alexander Taubin and Dr. Lev Levitin of Boston University whose insights and discussions were invaluable to the ideas of this paper.

## 12. REFERENCES

[1] M.G. Karpovsky and P. Nagvajara, "Optimal codes for the minimax criteria on error detection," *IEEE Trans. on Information Theory*, vol. 35, no. 6, pp. 1299–1305, 1989.

[2] M.G. Karpovsky and P. Nagvajara, "Optimal robust compression of test response," *IEEE Trans. on Computers*, vol. 39, no. 1, pp. 138–141, 1990.

[3] M.G. Karpovsky and Alexander Taubin, "New class of nonlinear systematic error detecting codes," *IEEE Trans. on Information Theory*, vol. 50, no. 8, pp. 1818–1820, 2004.

[4] M.G. Karpovsky, Konrad J. Kulikowski, and Alexander Taubin, "Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard," *dsn*, vol. 00, pp. 93, 2004.

[5] Claude Carlet and Cunsheng Ding, "Linear codes from perfect nonlinear mappings and their secret sharing schemes," *IEEE TRANSACTIONS ON INFORMATION THEORY*, vol. 51, no. 6, 2005.

[6] M.G.Karpovsky, R.S.Stankovic, and J.T.Astola, *Spectral Logic and Its Applications in Design of Digital Devices*, John Wiley & Sons, to be published in 2008.

[7] A.N. Degtyaryov and V.G. Slyozkin, "Conception of a generalized receiver in telecommunication systems," in *Proc. 11th Int. Conf. Microwave and Telecommunication Technology*, 2001, pp. 290–291.

[8] H. Sherman, "Some optimal signals for time measurement," *Trans IRE*, vol. IT-2, no. 1, 1956.

[9] F.H. Lange, "Correlation techniques," Princeton, N.J., Van Nostrand, 1967.

Table 3. Table of Perfect and Optimum Robust Codes

| Parameters | Construction Method | Perfect | Optimum |
|---|---|---|---|
| $(1, \frac{q^m-1}{q-1}, \frac{q^{m-1})}{q-1})_{\frac{q^{m+1}}{q-1}}, m \geq 2$ is an integer | Singer Difference Set | Yes | Yes |
| $(1, 2m-1, m-1)_{4m-1=q}$ | Paley Difference Set | Yes | Yes |
| $(1, m^2, \frac{(m^2-1)}{4})_{4m^2+1=q}$, m is an odd positive integer | Biquadric Residue Difference Sets | Yes | Yes |
| $(2sr, q^{(2s-1)r} + (q^r-1)q^{(s-1)r}, q^{(2s-2)r} + (q^r-1)q^{(s-1)r})_q$ | Quadratic Nonsystematic Code $\sigma = 0$ | q=2,r=1 | q=2,r=1 |
| $(2sr, q^{(2s-1)r} - q^{(s-1)r}, q^{(2s-2)r} \pm q^{(s-1)r})_q$ | Quadratic Nonsystematic Code $\sigma \neq 0$ | q=2,r=1 | q=2,r=1 |
| $((2s+1)r, q^{2sr}, q^{(2s-1)r})_q$ | Quadratic Systematic Code | No | Yes |
| $((2s+1)r, q^{2sr} + q^r - 1, q^{(2s-1)r} + 2)_q$ | Augmented Quadratic Systematic Code | No | s=1 |
| $((2s+1)r - b, q^{2sr}, q^{(2s-1)r+b})_q$ | Punctured Quadratic Systematic Code | No | Yes |
| $(2r, q^r, 1)_q, q \neq 2$ | Nonbinary Robust Duplication Code | No | Yes |
| $(2r-b, q^r, q^b)_q, q \neq 2$ | Nonbinary Punctured Robust Duplication Code | No | Yes |
| $(2r, 2^r, 2)_2$ | Binary Robust Duplication Code | No | No |

*In this table q is a prime power. For the construction of quadratic nonsystematic code, please refer to [1].


[10] Dieter Jungnickel and Alexander Pott, *Difference sets, sequences and their correlation properties*, chapter Difference sets: An introduction., Kluwer Academic Publishers, 1999.

[11] T. Beth, D. Jungnickel, and H Lenz, *Design Theory*, Cambridge University Press, 1999.

[12] H. S. M. Coxeter, *The Real Projective Plane, 3rd ed*, Springer Verlag, 1995.

[13] Claude Carlet and Cunsheng Ding, "Highly nonlinear mappings," *J. Complex.*, vol. 20, no. 2-3, pp. 205–244, 2004.

[14] P.K. Lala, *Self-Checking and Fault-Tolerant Digital Design*, Morgan Kaufman, 2001.

[15] Jin Yuan, C. Carlet, and Cunsheng Ding, "The weight distribution of a class of linear codes from perfect nonlinear functions," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 712–717, 2006.

[16] Mandi S. Maxwell, *Almost Perfect Nonlinear functions and related combinatorial structures*, Ph.D. thesis, ISU, 2005.

[17] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, NORTH-HOLLAND, 1983.

[18] I. Bouyukliew and Z. Varbanov, "Some results for linear binary codes with minimum distance 5 and 6," *Information Theory, IEEE Transactions on*, vol. 51, no. 12, pp. 4387–4391, 2005.

[19] J.F. Ziegler and H.Puchner, "SER history, trends, and challenges, a guide for designing with memory ICs," Cypress Semiconductor Corporation, 2004.

[20] K. Chakraborty and P. Mazumder, *Reliability and Fault Tolerance of RAMs*, Prentice Hall, 2002.

[21] D. Boneh, R. DeMillo, and R. Litpton, "On the importance of eliminating errors in cryptographic computations," *Journal of Cryptology*, vol. 14, no. 2, pp. 101–119, 2001.

[22] M.G. Karpovsky, L. Levitin, and A. Trachtenberg, "Data verification and reconciliation with generalized error control codes," *IEEE Trans. on Information Theory*, vol. 49, no. 7, pp. 1788–1794, 2003.