On-line Self Error Detection with Equal Protection Against All Errors

Mark G. Karpovsky, Konrad J. Kulikowski, Zhen Wang
(markkar@bu.edu, konkul@bu.edu, lark@bu.edu)

Reliable Computing Laboratory
Boston University
8 Saint Mary's Street, Boston, MA 02215, USA

Abstract:
Error characteristics for many communication and computation channels are non-stationary or are difficult to determine. In such cases, traditional minimum distance codes do not minimize the worst case error masking probabilities. We present *robust* codes which minimize the maximum error masking probability when the error distributions are unknown. The properties of these robust codes as well as optimum constructions for a wide range of practical parameters are analyzed.

We discuss applications of these codes for reliable transmission and storage. We also present several design techniques for memories with self-error-detection based on the proposed codes and robust check-point verification providing equal or almost equal protection against all errors. The proposed techniques are efficient when error vectors at the outputs of the protected devices (which may be results of internal stuck-at faults) have a high probability of repeating themselves. The proposed robust codes require slightly larger overhead than standard and widely-used linear codes but provide for drastic increase in performance (probability of detection).

1 Introduction

Standard methods for on-line error detection are based on simple linear error-detecting codes. Typical methods include the use of parity, replication, or Hamming codes [1]. These traditional linear error detection methods are best suited for detection of errors of a small multiplicity. The codes concentrate their error-detecting power on a small class of errors which are assumed most likely to be the observed manifestations of faults and failures. For many traditional linear codes such as parity or Hamming the most probable errors are assumed to be single or double bit distortions.

However, for many computational devices and many environments the assumptions which make the traditional methods efficient cannot be guaranteed. In most complex computational devices and many environments even single faults do not result in single or double errors at the output but in the corruption of several bits. Due to the complexity of devices the exact probabilities of different error patterns at the outputs of the devices to be protected are difficult to determine. Furthermore, increased scaling of device feature sizes are causing increased probabilities of multibit faults. In the presence of non-stationary or multibit faults and errors the existing methods can exhibit poor or very difficult to predict error detection characteristics making it difficult to establish reliability guarantees for the protection methods based on linear codes.

In light of these limitations alternate methods based on nonlinear codes known as robust codes are proposed in this paper. Robust codes are designed to provide uniform error detection against all errors instead of concentrating their error-detecting power on a specific class of errors. Nonsystematic robust codes were introduced in [2] and first applied to the compression of test responses [3]. In [4] systematic constructions of robust codes were first proposed and applied to the protection of cryptographic hardware against fault attacks in [5]. In this paper, we outline the properties and constructions for optimum binary systematic robust codes and provide examples of applications of robust error-detecting codes for on-line error detection for lazy computation channels and memories as well as check-point verification.

2 Definition and Basic Properties of Robust Codes

In this paper we mainly consider applications of codes for error detection in digital hardware. We therefore present all of the definitions in terms of binary codes. Most of the results can be easily generalized for codes over nonbinary fields.

Definition 2.1 ***(R-Robust Code)*** *A code $C \subseteq GF(2^n)$ is **R-robust** if the size of the intersection of the code C and any of its translates $\tilde{C} = \{\tilde{w}|\tilde{w} = w + e, w \in C, e \in GF(2^n), e \neq 0\}$ is upper bounded by R:*

$$R = \max_{0 \neq e \in GF(2^n)} |\{w|w \in C, w + e \in C\}|. \tag{1}$$

where $+$ is the componentwise addition modular two. A binary R-robust code C of length n with $M = |C|$ is denoted by a triple (n, M, R).

The above defined robust codes have beneficial properties when worst case error masking probability of the codes is considered. By definition of a R-robust code there are at most R codewords which can mask any fixed error e. The error masking probability $Q(e)$ can be thus defined as

$$Q(e) = \frac{|\{w|w \in C, w + e \in C\}|}{|C|}. \tag{2}$$

For a (n, M, R) robust code, the worst case probability of masking an error is at most R/M for any error when the codewords of the robust code are assumed equiprobable. Clearly, robust codes which have a minimum R for a given M will also have the lowest probability of error masking and hence a predictable behavior in the presence of unpredictable error distributions since the worst case probability of masking any error is bounded. In the following sections we investigate the constructions and optimality of the codes followed by some examples of applications.

3 Bounds, Optimality, and Perfect Robust Codes

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

Property 3.1 *If the code C is R-robust then in the multiset $S_C = \{w_j + w_i | w_i, w_j \in C, w_i \neq w_j\}$, any element appears at most R times.*

Robust codes are optimal if they have the maximum number of codewords M for a given R and and length n. From Property 3.1, a relation on R, n and M of the code can be established.

$$M^2 - M \leq R(2^n - 1). \tag{3}$$

Definition 3.1 ***(Perfect Robust Code)*** *A robust (n, M, R) code satisfying $M^2 - M = R(2^n - 1)$ is **perfect**.*

Perfect robust codes are equivalent to classical combinatorial structures known as difference sets and symmetric designs [6]. It has been shown by Mann that all symmetric designs over binary fields and hence perfect binary robust codes exist only for even dimensions and are limited to the following parameters: $(2m + 2, 2^{2m+1} \pm 2^m, 2^{2m} \pm 2^m)$ [7]. Moreover, *systematic* robust codes, which are often more practical for error detection in computer hardware due to their separation of data and check bits, cannot be perfect.

Theorem 3.1 *For any $(n, 2^k, R)$ systematic robust code there are at least 2^{n-k} in $GF(2^n)$ elements which cannot be expressed as differences of two codewords.*

Proof For any systematic codeword $w = (x_1, x_2 = f(x_1))$ an error $e = (y_1, y_2)$ is masked iff $f(x_1 + y_1) = x_2 + y_2$. An error $e = (y_1 = 0, y_2 \neq 0)$ is never masked since $f(x_1) = x_2 + y_2$ only iff $y_2 = 0$. An error that is never masked cannot be expressed as a difference of two codewords. Hence elements from $GF(2^n)$ of the form $w = (0, x_2 \in GF(2^r))$, where $r = n - k$ cannot be expressed as a difference of two codewords.

Corollary 3.1 *There are no perfect systematic robust codes.*

When perfect robust codes are not available, the best possible codes which maximize M for a given n and R are referred to as *optimum* robust codes.

Definition 3.2 ***(Optimum Robust code)*** *(n, M, R)robust codes which have the maximum possible number of codewords M for a given length n and robustness R with respect to (3) are called **optimum**. For optimum codes adding any additional codewords would violate bound (3) and*

$$M^2 - M \leq R(2^n - 1) < M^2 + M. \tag{4}$$

4 Constructions of Optimal Systematic Robust Code

There is a strong relationship between robust codes, nonlinearity, and nonlinear functions since all robust codes are nonlinear. The parameters of systematic robust codes depend on nonlinearity of the encoding function of the codes.

We first review some basic definitions and properties of nonlinearity, a good survey of nonlinear functions can be found in [8].

Let f be a function that maps elements from $GF(2^k)$ to $GF(2^r)$.

$$f : GF(2^k) \rightarrow GF(2^r) : a \rightarrow b = f(a). \tag{5}$$

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

$$P_f = \max_{0 \neq a \in GF(2^k)} \max_{b \in GF(2^r)} Pr(D_a f(x) = b), \tag{6}$$

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

Definition 4.1 *A binary function $f : GF(2^k) \rightarrow GF(2^r)$ has **perfect nonlinearity** iff $P_f = \frac{1}{2^r}$.*

Theorem 4.1 *Let f be a nonlinear function that maps $GF(2^k)$ to $GF(2^r)$ 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 $x_1 \in GF(2^k)$ and $x_2 \in GF(2^r)$ forms a $(k + r, 2^k, 2^k P_f)$-robust systematic code.*

Proof The error $e = (y_1, y_2), (y_1 \in GF(2^k), y_2 \in GF(2^r))$ will be masked iff $f(x_1 + y_1) + f(x_1) = y_2, x_1 \in GF(2^k)$, which is exactly when $D_{y_1} f(x_1) = y_2$.

The following is an example of a construction based on perfect nonlinear functions resulting in optimum robust codes.

Construction 4.1 ***(Quadratic Systematic Code)*** *Let $w = (x_1, x_2, \cdots, x_{2s}, x_{2s+1})$, $x_i \in GF(2^r), s \geq 1$. A vector $w \in GF(2^{(2s+1)r})$ belongs to the code iff*

$$x_1 x_2 + x_3 x_4 + \cdots + x_{2s-1} x_{2s} = x_{2s+1}. \tag{7}$$

The resulting code is a $((2s + 1)r, 2^{2sr}, 2^{(2s-1)r})$ 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 $P_f = 1/2^r$ [8]. For $r = 1$ this function is known as a *bent* function. From Theorem 4.1 the resulting code is a $R = 2^{2sr}/2^r = 2^{(2s-1)r}$ robust code.

We will discuss applications of these quadratic codes for designing of memories with self-error-detection in Section 6.1.2, for data transmission in noisy channel in Section 6.2 and for data verification in Section 6.3.

Example 4.1 ***(Robust Parity)*** *Methods based on linear parity check codes are often used for on-line error detection in combinational circuits [9]. The linear 1-dim 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 1-dim parity codes can result in unpredictable behavior in the presence of errors.*

As an alternative to the 1-dim parity codes the quadratic systematic robust codes defined in Construction 4.1 can be used. Taking $r = 1$ the resulting systematic code has the same redundancy as the linear parity code. Unlike the linear parity code, the robust code will mask an error with a probability of at most $\frac{1}{2}$ regardless of the error multiplicity providing predictable error detection regardless of the error distributions.

Perfect nonlinear functions from $GF(2^k)$ to $GF(2^k)$ do not exist [8]. Functions with optimum nonlinearity in this case are called almost perfect nonlinear (APN) functions [10] with $P_f = 2^{-k+1}$. When f are APN functions, the robust codes constructed as in theorem 4.1 have $R = 2$. These codes are not optimum.

Construction 4.2 ***(Robust Duplication Code)*** *Let $w = (x_1, x_2), x_1, x_2 \in GF(2^r)$. The robust duplication code C contains all vectors $w \in GF(q^{2r})$ which satisfy $x_1^3 = x_2$ where all the computations are in $GF(2^r)$. The code is a $(2r, 2^r, 2)$ robust code.*

Robust duplication codes can be a viable alternative to standard duplication techniques. Application of these codes to memories with self-error detection and comparison between standard and robust duplication techniques will be discussed in Section 6.2.

5 Partially Robust Codes

Robust codes generally have higher complexity of encoding and decoding than classical linear codes. The $((2s + 1)r, 2^{2sr}, 2^{(2s-1)r})$ quadratic systematic codes from Construction 4.1 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 the 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 [4]. These codes combine linear and nonlinear mappings to decrease the hardware overhead associated with generation of check bits by the predictor. The encoding of systematic partially robust code is performed first by using a linear function to compute the redundant r check bits followed by nonlinear transformation. The use of the linear code as the first step in the encoding process typically results in a hardware savings in the encoder or predictor since the nonlinear function needs to only be computed based on the r bit output of the linear block. The application of the nonlinear transformation reduces the number of undetectable errors thus increasing the robustness of the linear codes.

Construction 5.1 ***(Partially Robust Codes)*** *Let $f : GF(2^r) \rightarrow GF(2^r)$ be a nonlinear function with $P_f < 1$ and let $H : GF(2^k) \rightarrow GF(2^r)$, $r \leq k$ be a linear onto function. The set of words in the form $(x, f(H(x)))$ form a code with 2^{k-r} undetectable errors.*

Proof By Theorem 4.1 the set of words in the form $(H(x), f(H(x)))$ forms a $(2r, 2^r, P_f 2^r)$ systematic robust code. For this code the only undetectable error is the zero error. Since H is a linear onto function it maps 2^{k-r} words from $GF(2^k)$ to the r-bit zero vector. Hence the code with words in the form $(x, f(H(x)))$ has a total of 2^{k-r} undetectable errors. (Matrix H can be selected as a check matrix of the best $[k + r, 2^k]$ linear code.)

The number of undetectable errors is reduced from 2^k to 2^{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 ins Section 6.1.1. Partially robust codes with $k = 128$ and $r = 32$ have been used in [5] for design of private key security devices on Advanced Encryption Standard (AES) resistant to fault injection attacks. Implementation of this approach resulted in about 80% hardware overhead.

6 Applications

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.

6.1 Self Error Detection for 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 [11]. 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 [12].

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 [11].

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

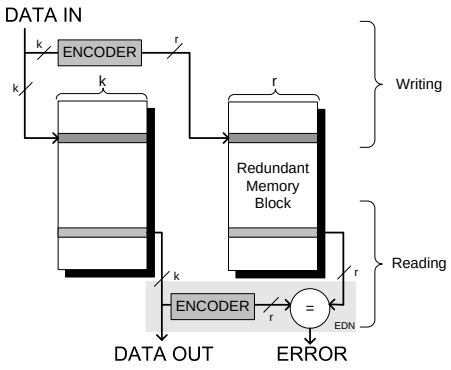


Figure 1: High level memory architecture with self error detection

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.

6.1.1 Existing Methods for Self Error Detection in Memories

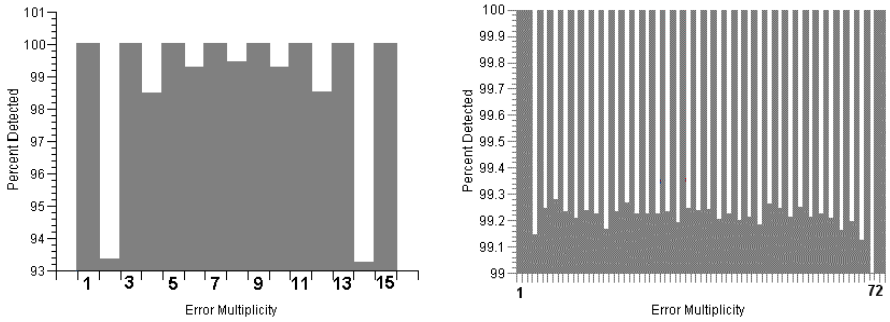
Two commonly used error detection schemes used in memories today are duplication and the extended Hamming codes [11]. Both of these schemes are based on very simple linear codes to reduce the hardware and time overheads.

The high level architecture for protected memory is shown in Figure 1. 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).

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 2 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

$[72, 2^{64}]$ 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.



(a) (b)

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

6.1.2 Robust Methods for Self-Error Detection in Memories

Figure 3 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 3 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 3a shows the error detection profile for a $(16, 2^8, 2)$ 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 subset of errors is chosen for this robust code the error masking probability is bounded by 2^{-7}.

Figure 3b shows the error detection profile for a $(72, 2^{64}, 2^{56})$ 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 3c 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 3b).

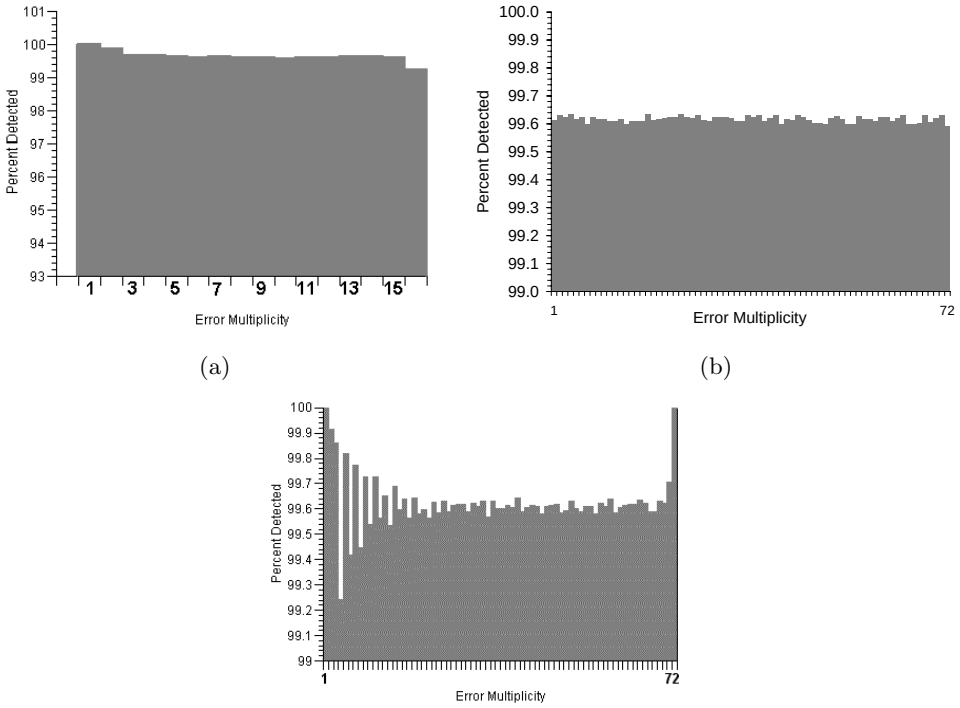


Figure 3: Percentage of errors detected for (a) $(16, 2^8, 2)$robust duplication code (b) $(72, 2^{64}, 2^{56})$ robust quadratic code (c) partially robust $(72, 2^{64})$ extended Hamming code

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 $[128, 2^{64}]$ linear duplication code and the corresponding $(128, 2^{64}, 2)$ robust code and between the linear $[72, 2^{64}]$ 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.

Table 1: Probability of Masking for linear and robust codes with $k = 64$s

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

6.2 Error Detection in Channels with Laziness

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

$$L(e) = Pr(e = \tilde{w}_{i+1} - w_{i+1} | e = \tilde{w}_i - w_i), \tag{8}$$

where $w_i \neq w_{i+1}, e \neq 0$ and all messages w are considered equiprobable.

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 [13]. 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 [5].

Example 6.1 ***(Circuit errors in combinational networks resulting from single stuck-at faults)***

Consider the circuit in Figure 4. 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

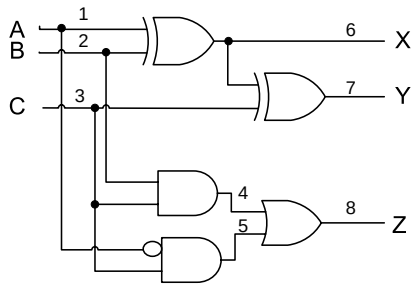


Figure 4: *A circuit with laziness*

Table 2: *Laziness of errors $L(e)$ for circuit from Figure 4*

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*

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 $(n, 2^k, R)$ 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 5 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

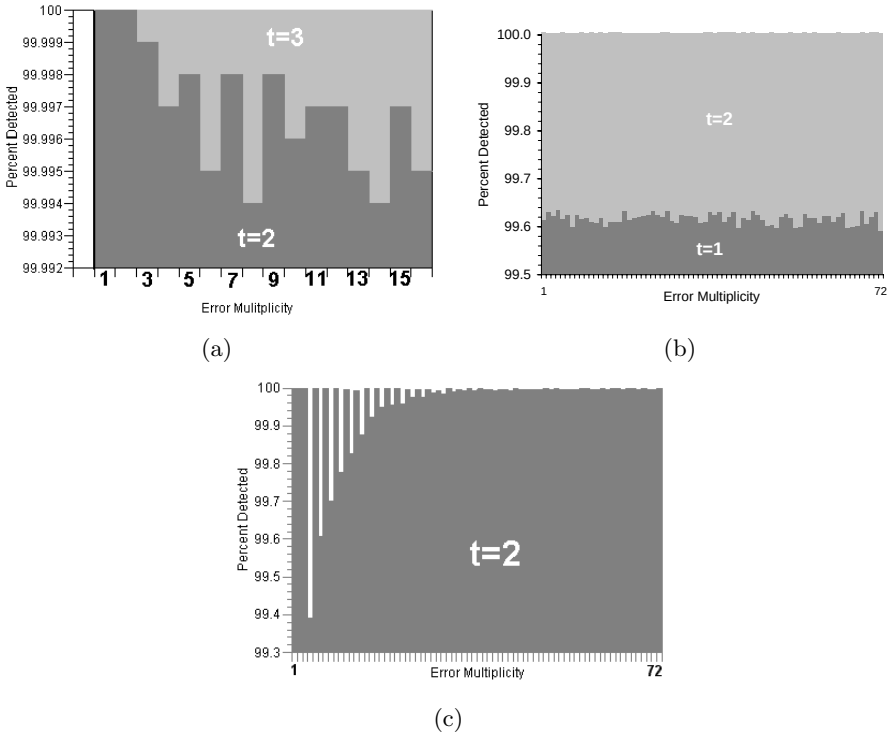


Figure 5: Percentage of errors detected for (a) 8-bit robust duplication after t=2 and 3 messages (b) $(72, 2^{64}, 2^{56})$ robust code (c) partially Robust Hamming code after t=2 data

different data being affected by the same error. For robust duplication, as is shown in Figure 5a, 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 $(72, 2^{64}, 2^{56})$ robust code is shown in Figure 5b. After two messages the probability of masking an error is reduced to 2^{-16} for any error.

Likewise, Figure 5c shows the probability of detecting an error for the partially robust $(72, 2^{64})$ 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 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.

6.2.1 Failure Detection Probability

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

$$Q(e, L(e)) = Q(e)(1 - L(e)) \sum_{i=0}^{\infty} L(e)^i Q(e)^i \tag{9}$$

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

$$Q(e, L(e)) = Q(e)(1 - L(e)) + \frac{Q(e)^2(1 - L(e))L(e)}{1 - Q(e)L(e)}. \tag{10}$$

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.

6.2.2 Failure Detection Latency

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 prob-

ability 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 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.

6.3 Robust Check-Point Verification

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 [14].

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 $(k + r = n, 2^k, R)$ 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 6.1 *For robust verification of a k bit data set based on a systematic $(n, 2^k, R)$ 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 4.1 any $(n, 2^k, R)$ systematic Robust code can be defined by the nonlinearity of its encoding function. For a systematic $(n, 2^k, R)$ 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 6.2 *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 4.1) 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 $(2^{24} + 64, 2^{2^{24}}, 2^{2^{24}-64})$ 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.*

7 Conclusions

In this paper we describe a new class of error detecting codes and apply them for error detection in communications, storage and for robust check-point verification. As opposed to the approaches based on standard linear codes the proposed approach provides for equal protection against all possible errors. This may be very useful when errors at the outputs of the channel or device we are protecting are difficult to predict. The proposed approach is most efficient for repeating errors. In this case it was shown that the proposed approach is more efficient than the approaches based on linear codes. For example, the proposed

robust codes with $n = 72$ or $n = 128$ and $k = 64$ are shown to be viable alternatives to widely used $[72, 2^{64}, 4]$ extended Hamming code or $[128, 2^{64}, 2]$ duplication code for the design of memories with self-error detection. For check-point verification robust quadratic codes with $r = 64$ are shown to be viable alternatives to the corresponding linear code.

8 Acknowledgment

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.

References

[1] I. Koren and C. M. Kirshna, *Fault-Tolerant Systems*. Elsevier, 2007.

[2] 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.

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

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

[5] M. Karpovsky, K. J. Kulikowski, and A. Taubin, "Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard," in *Proceedings of Dependeable Systems and Networks*, 2004.

[6] D. Jungnickel and A. Pott, *Difference sets, sequences and their correlation properties*, ch. Difference sets: An introduction. Kluwer Academic Publishers, 1999.

[7] T. Beth, D. Jungnickel, and H. Lenz, *Design Theory, Volume 1*. Cambridge University Press, 1999.

[8] C. Carlet and C. Ding, "Highly nonlinear mappings," *Journal of Complexity*, vol. 20, no. 2-3, pp. 205–244, 2004.

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

[10] M. S. Maxwell, *Almost Perfect Nonlinear functions and related combinatorial structures*. PhD thesis, ISU, 2005.

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

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

[13] 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.

[14] M. 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.