Nonlinear Multi-Error Correction Codes for Reliable NAND Flash Memories

Zhen Wang, *Student Member, IEEE,* Mark Karpovsky, *Fellow, IEEE,* and Ajay Joshi, *Member, IEEE*

***Abstract*—Multi-level cell (MLC) NAND flash memories are popular storage media because of their power efficiency and big storage density. Conventional reliable MLC NAND flash memories based on BCH codes or Reed-Solomon codes have a large number of undetectable and miscorrected errors. Moreover, standard decoders for BCH and Reed-Solomon codes cannot be easily modified to correct errors beyond their error correcting capability $t = \lfloor \frac{d-1}{2} \rfloor$, where d is the Hamming distance of the code. In this paper, we propose two general constructions of nonlinear multi-error correcting codes. The proposed constructions can generate nonlinear bit-error correcting or digit-error correcting codes with very few or even no errors undetected or miscorrected by all codewords. Moreover, codes generated by the generalized Vasil'ev construction can correct some errors with multiplicities larger than t without any extra overhead in area, latency and power consumption compared to schemes where only errors with multiplicity up to t are corrected. The design of reliable MLC NAND flash architectures can be based on the proposed nonlinear multi-error correcting codes. The reliability, area overhead and the penalty in latency and power consumption of the architectures based on the proposed codes are compared to architectures based on BCH codes and Reed-Solomon codes. The results show that using the proposed nonlinear error correcting codes for the protection of MLC NAND flash memories can reduce the number of errors undetected or miscorrected by all codewords to be almost 0 at the cost of less than 20% increase in power and area compared to architectures based on BCH codes and Reed-Solomon codes.**

I. Introduction

The semiconductor industry has witnessed an explosive growth of the NAND flash memory market in the past several years. Due to its high data transfer rate, low power consumption, large storage density and long mechanical durability, the NAND flash memories are widely used as storage media for devices such as portable media players, digital cameras, cell phones and low-end netbooks.

The increase of the storage density and the reduction of the cost per bit of flash memories were traditionally achieved by the aggressive scaling of the memory cell transistor until the multi-level cell (MLC) technology was developed and implemented in 1997 [1]. MLC technology is based on the ability to precisely control the amount of charge stored into the floating gate of the memory cell for the purpose of setting the threshold voltage to a number of different levels corresponding to different logic values, which enables the storage of multiple bits per cell.

However, the increased number of programming threshold voltage levels has a negative impact on the reliability of the device due to the reduced operational margin. The raw bit error rate of the MLC NAND flash memory is around 10^{-6} [2] and is at least two orders of magnitude worse than that of the single-level cell (SLC) NAND flash memory [3]. Moreover, the same reliability concerns as for SLC NAND flash memories, e.g. program/read disturb, data retention, programming/erasing endurance [4] and soft errors [5][6][7], may become more significant for MLC NAND flash memories. Hence a powerful error correcting code (ECC) that is able to correct at least 4-bit errors is required for the MLC NAND flash memories to achieve an acceptable application bit error rate, which is no larger than 10^{-11} [2].

Several works have investigated the potential usage of linear block codes to improve the reliability of MLC NAND flash memories. In [8], the authors presented a high-throughput and low-power ECC architecture based on $(n = 4148, k = 4096, d = 9)$ BCH codes correcting quadruple errors ($t = 4$). In [9], a 4Gb 2b/cell NAND flash memory chip incorporating a 250MHz BCH error correcting architecture was shown. The author of [10] demonstrated that the use of strong BCH codes (e.g. $t = 12, 15, 67, 102$) can effectively increase the number of bits/cell thus further increase the storage capacity of MLC NAND flash memories. In [11], an adaptive-rate ECC architecture based on BCH codes was proposed. The design had four operation modes with different error correcting capabilities. An ECC architecture based on Reed-Solomon codes of length 828 and 820 information digits constructed over $GF(2^{10})$ was proposed in [12], which can correct all bit errors of multiplicity less or equal to four. The architecture achieves higher throughput, requires less area overhead for the encoder and the decoder but needs 32 more redundant bits than architectures based on BCH codes with the same error correcting capability. In [13], an architecture based on asymmetric limited-magnitude error correcting code was proposed, which can correct all asymmetric errors of multiplicities up to t.

The above architectures are based on linear block codes and have a large number of undetectable errors. Indeed, for any linear code with k information bits, the number of undetectable errors is 2^k, which is a potential threat to the reliability of the memory systems. The situation becomes even worse due to the possible miscorrection of errors. Denote a binary error vector by e and the multiplicity of the error by $||e||$. A multi-bit error $e, ||e|| > t$ is miscorrected by a linear t-error-correcting code if and only if it has the same syndrome as some e', where $||e'|| \leq t$. It is easy to show that the number of errors miscorrected by all codewords of a (n, k, d) linear t-error-correcting code is $\sum_{i=1}^{t} \binom{n}{i} \times (2^k - 1)$.

Zhen Wang, Mark Karpovsky and Ajay Joshi are with the Department of Electrical and Computer Engineering, Boston University. The work of the second author is supported by the NSF grant CNR 1012910.

Under the assumption that errors are independent whose distribution satisfies $P(e) = \theta^{||e||}(1-\theta)^{n-||e||}$, where θ is the raw bit distortion rate and $P(e)$ is the probability of the occurrence of event e, the most harmful miscorrected errors are errors of multiplicity $t+1$. Let us denote the number of codewords of Hamming weight $2t+1$ for a linear error correcting code by A_{2t+1}. The number of errors of multiplicity $t+1$ that are miscorrected by all codewords of a linear bit-error correcting code is $\binom{2t+1}{t} \times A_{2t+1}$. For the commonly used $(n = 8262, k = 8192, d = 11)$ linear BCH codes with $t = 5$, the number of errors of multiplicity six miscorrected by all codewords is as large as $462 \times A_{11} \approx 10^{17}$. This large number of miscorrected errors of multiplicity six can not be neglected and should be taken into account when designing reliable MLC NAND flash memories.

To reduce the number of undetectable and miscorrected errors, nonlinear *minimum distance robust* and *minimum distance partially robust* codes have been proposed in [14], [15], [16]. An ECC architecture based on nonlinear single-error-correcting, double-error-detecting (SEC-DED) codes for the protection of memories against soft errors has been shown in [15].

The contribution of this paper is threefold. First, we present two general constructions of nonlinear multi-error correcting codes in $GF(p^n)$, where n is the length of the code and p is a power of prime. The first construction is based on the idea of concatenating linear and nonlinear redundant digits. It can generate nonlinear robust codes with no undetectable errors and no errors miscorrected by all codewords at the cost of extra redundant digits compared to BCH codes and Reed-Solomon codes with the same error correcting capabilities. The second construction is generalized from the existing nonlinear perfect Hamming codes, i.e. Vasil'ev codes [17]. These codes are partially robust codes and can be as good as BCH codes and Reed-Solomon codes in terms of the number of redundant digits but have less undetectable errors and errors miscorrected by all codewords.

Second, we present the error correcting algorithms for both constructions of nonlinear multi-error correcting codes. The error correcting algorithm for the generalized Vasil'ev codes can also correct some errors beyond the error correcting capability t without any modifications and extra requirements. When $p = 2^m$, the presented constructions can also generate nonlinear error correcting codes with the same digit-error and burst-error correcting capabilities as Reed-Solomon codes in $GF(2^m)$.

In addition to errors that are undetectable or miscorrected by all codewords, there are also some errors which are masked or miscorrected by a fraction of codewords of the presented nonlinear multi-error correcting codes. These errors are called **conditionally detectable** and **conditionally miscorrected** errors. We note that the data-dependent error detecting and correcting properties of the presented codes are useful for detecting and locating repeating errors, e.g. errors introduced by hardware malfunctions such as data retention and programming/erasing failure.

Third, we propose ECC architectures for MLC NAND flash memories based on the presented nonlinear multi-error correcting codes. The proposed architectures have nearly no undetectable errors and errors miscorrected by all codewords at the cost of less than 20% increase in area and power consumption compared to architectures based on BCH and Reed-Solomon codes with the same bit error correcting capability t. Moreover, the reliability of the memories protected by the generalized Vasil'ev codes can be further improved given the fact that the proposed architecture is able to correct some errors of multiplicity larger than t.

The rest part of the paper is organized as follows. In Section II, we briefly review the architecture of MLC NAND flash memories and explain the error model we use in the paper. In Section III, the definitions of robust codes and partially robust codes are given. In Section IV, the two general constructions of nonlinear multi-error correcting codes will be shown. As opposed to the known nonlinear perfect Hamming codes, the presented constructions can generate nonlinear codes with any given length and Hamming distance. The error correcting algorithm for the proposed nonlinear multi-error correcting codes will be described and the error correcting capabilities of these codes will be analyzed and compared to BCH codes and Reed-Solomon codes. In Section VI, the hardware design of the encoder and the decoder for the nonlinear multi-error correcting codes will be given. The area overhead, the latency and the power consumption of the design will be estimated and compared to designs based on BCH codes and Reed-Solomon codes.

This paper is an extended version of our work presented in [18].

II. MLC NAND Flash Memories

Multi-level cell is able to store multiple bits by precisely controlling the threshold voltage level of the cell. In practice, the threshold voltage of the whole memory array satisfies a Gaussian distribution due to random manufacturing variations. Figure 1 illustrates the threshold voltage distribution of a multi-level cell which can store 2 bits. Let us denote the standard deviation of the middle two Gaussian distributions in Figure 1 by σ. The standard deviations of the outer two distributions are approximately 4σ and 2σ [12]. Each voltage range corresponds to a specific logic value represented as a 2-bit binary vector. Different schemes can be used for mapping the logic values to binary vectors. A direct mapping was used in [1]. The authors in [12] proposed to use a Gray mapping to improve the reliability of the memory. If an error occurs during a READ operation, it is more likely that the original 2-bit binary vector is distorted into another 2-bit vector corresponding to the adjacent voltage level (Figure 1). Hence the Gray mapping can efficiently reduce the average error multiplicity thus increasing the error detecting probability compared to the direct mapping scheme.

The data of the NAND flash memory is organized in *blocks* (Figure 2). Each block consists of a number of *pages*. Each page stores K *data bytes* and R *spare bytes*. The spare area is physically the same as the rest of the page and is typically used for overhead functions such as ECC and wear-leveling [19]. The proportion of the spare bytes in the total number of

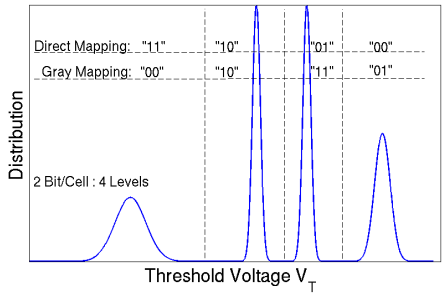


Fig. 1: Threshold voltage distribution for a MLC storing 2 bits [12]

bytes per page is usually 3% , e.g. 64 spare bytes for 2048 data bytes. More spare bytes may be required as the page size increases, e.g. 218 spare bytes for 4096 data bytes [2]. Due to the existence of spare bytes, the number of redundant bits of the error correcting codes used for NAND flash memories is not as critical as for other types of memories such as SRAM and DRAM where the area overhead is mostly determined by the number of redundant bits. This allows for a flexible design of more powerful error correcting codes for NAND flash memories.

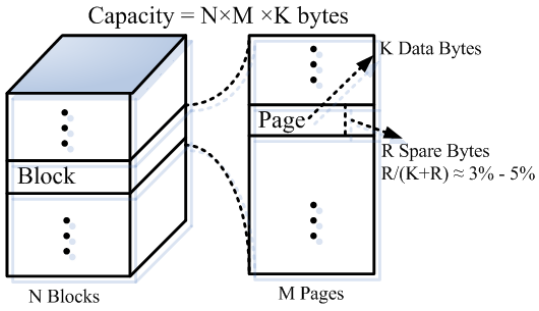


Fig. 2: Data organization in NAND flash memories

Similar to SLC flash memories, the primary failure mechanisms for MLC NAND flash memories include threshold voltage distribution, program/read disturb, data retention, programming/erasing endurance and single event upset. However, while for SLC flash memories a lot of errors are asymmetric, e.g. errors introduced by program disturb and data retention [13], for MLC NAND flash memories errors have no preferred symmetry [20]. Moreover, experimental results show that errors in MLC flash memories are more likely to occur uniformly within a page without any observable burstiness or local data dependency [20]. Thereby, throughout the paper we assume a random symmetric error model. Let c be the error-free output of the memory and e be the error vector. The distorted output $\tilde{c}$ can be written as $c \oplus e$, where $\oplus$ is the XOR operation. The probability of a nonzero error e can be computed as $P(e) = \theta^{||e||}(1-\theta)^{n-||e||}$, where θ is the raw bit distortion rate and $||e||$ is the multiplicity (the number of nonzero components) of the error. We want to emphasize that the proposed nonlinear multi-error correcting codes not only have advantages over linear codes under this error model but can also provide a guaranteed level of reliability in situations where the error model is unpredictable or multi-bit errors are more probable.

III. Definitions of Nonlinear Robust and Partially Robust Codes

Throughout the paper we denote the addition in Galois field $GF(p)$ by $\oplus$ and denote an error control code with length n, dimension k and Hamming distance d by (n, k, d). In general, the error correcting capability is $t = \lfloor \frac{d-1}{2} \rfloor$.

Definition 1: (**Kernels of the code**) For any error correcting code $C \subseteq GF(p^n)$, the **detection kernel** K_d is the set of errors that are masked by all codewords.

$$K_d = \{e | e \oplus c \in C, \forall c \in C\}. \quad (1)$$

It is easy to show that K_d is a linear subspace of C. If C is linear, $K_d = C$.

Let us denote the error correction algorithm for code C by A and denote the set of errors that A attempts to correct by E. The **correction kernel** K_c is defined as follows:

$$K_c = \{e | e \notin E, \forall c \in C, \exists e^{'} \in E, A(e \oplus c) = A(e^{'} \oplus c)\}. \quad (2)$$

A main characteristic of traditional linear error detecting codes is that they concentrate their error detecting and correcting power on a small subset of errors which are assumed to be the most likely to occur. Typically, such codes concentrate their error detection on errors of a small multiplicity. They are designed to guarantee detection of all errors with a multiplicity less than d. Error detection beyond the minimum distance of the code is typically not a part of the design criteria and can be unpredictable and ineffective. While for some classes of errors the codes provide 100% protection, for a very large class of errors linear codes offer no protection for all messages. For linear codes, $K_d = C$, these codes have the largest detection kernel K_d (the set of undetectable errors) of any class of systematic codes with the same n and k.

Robust codes are designed to provide a guaranteed level of detection against all errors. These codes are characterized by their error masking probability $Q(e)$, which is the fraction of codewords that mask a given error e.

$$Q(e) = \frac{|\{c | c \in C, c \oplus e \in C\}|}{|C|}. \quad (3)$$

Definition 2: The code C is **robust** iff $\max_{e \neq 0} Q(e) < 1$, or equivalently the detection kernel of the code contains only the zero vector $K_d = \{0\}$.

Robust codes are optimal when the maximum $Q(e)$ for all errors is minimized [21]. For a robust code the error masking probability is bounded for nonzero errors. Most robust codes do not have a minimum distance larger than one and do not guarantee 100% detection probability for any subset of errors. A possible variant of the robust codes is to include a minimum distance into the design criteria.

Definition 3: Let $||e||$ denote the multiplicity of an error e. A robust code where $Q(e) = 0$ for all $||e|| < d, e \neq 0$ is a **d-minimum distance robust code**.

Minimum distance robust codes have no undetectable errors and the worst case error masking probability is bounded by

$\max_{e\neq 0} Q(e) < 1$. However, unlike traditional robust codes they also provide a guaranteed 100% probability of detection of errors of small multiplicities ($||e|| < d$). These codes can be useful for providing the highest protection against the most likely or most dangerous threat while maintaining a detection guarantee in case of an unexpected behaviour.

For some applications the error characteristics of robust codes can be considered too pessimistic. *Partially robust* codes and *minimum distance partially robust* codes (see Definition 4) allow for a tradeoff among robustness, decoding complexity and overheard, which fill the gap between the optimistic linear codes and pessimistic robust codes.

Definition 4: A (n, k, d) code with a detection kernel smaller than the size of the code $|C|$ is a **partially robust code**. If the code also has a minimum distance greater than one it is referred to as a **minimum distance partially robust code**.

Partially robust codes reduce the number of undetectable errors while preserving some structures of linear codes which can be exploited to build efficient prediction hardware that generates redundant bits of a message. Like linear codes, partially robust codes still have undetectable errors (hence they are not completely robust). The number of undetectable errors is reduced by many orders of magnitude compared to that of the linear codes. For practical partially robust constructions, the number of undetectable errors can be reduced from p^k to p^{k-r} compared to a linear p-ary (n, k, d) code [14].

In the next section, we present the general constructions of a robust and a partially robust nonlinear multi-error correcting codes. We describe the error correcting algorithms for these codes, compare their error detecting and correcting capabilities to BCH and Reed-Solomon codes for the protection of MLC NAND flash memories.

IV. Constructions of Nonlinear Multi-Error Correcting Codes

The error detecting properties of nonlinear codes are highly related to nonlinear functions. The nonlinearity P_f of a function $f : GF(p^k) \rightarrow GF(p^r)$ can be defined by (from [22])

$$P_f = \max_{0\neq a\in GF(p^k)} \max_{b\in GF(p^r)} P(f(x \oplus a) \oplus f(x) = b), \quad (4)$$

where $P(E)$ denotes the probability of occurrence of event E. The smaller the value of P_f is, the higher the corresponding nonlinearity of f is. When $P_f = p^{-r}$, f is a perfect nonlinear function.

A. Multi-Error Correcting Codes Based on Concatenations

The first construction of nonlinear multi-error correcting codes is based on the idea of concatenating linear and nonlinear redundant digits.

Theorem 1: Let $f : GF(p^k) \rightarrow GF(p^{r_n})$ be a nonlinear function with nonlinearity P_f. Let $V = \{(z, \phi(z))\}$ be a linear code with Hamming distance d, where $z \in GF(p^{k+r_n})$ and $\phi(z) : GF(p^{k+r_n}) \rightarrow GF(p^{r_l})$ is the encoding function. The code defined by

$$\{(y, f(y), \phi(z))\}, \quad (5)$$

where $y \in GF(p^k), f(y) \in GF(p^{r_n}), z = (y, f(y)) \in GF(p^{k+r_n})$ and $\phi(z) \in GF(p^{r_l})$ is a robust error correcting code with Hamming distance d and $K_d = K_c = \{0\}$. Any nonzero error will be detected with a probability of at least $1 - P_f$.

Proof: Let $e = (e_1, e_2, e_3)$ be the error vector, where $e_1 \in GF(p^k), e_2 \in GF(p^{r_n})$ and $e_3 \in GF(p^{r_l})$. The error masking equations can be written as:

$$f(y \oplus e_1) = f(y) \oplus e_2; \quad (6)$$

$$\phi(z + (e_1, e_2)) = \phi(z) + e_3. \quad (7)$$

1) If $e_1 = 0$ and e_2, e_3 are not both 0, at least one of the equations shown above will not be satisfied. The error will always be detected.
2) If $e_1 \neq 0$, from the definition of nonlinear functions, there are at most $P_f p^k$ solutions of y for (6). Thereby, the error will be masked with a probability of at most P_f.

■

The resulting code C in Theorem 1 has the same Hamming distance d and the same error correcting capability t as the linear code V. Moreover, the error correcting algorithm for V can be slightly modified to correct errors in C.

Let $e = (e_1, e_2, e_3)$ be the error vector, $c = (x_1, x_2, x_3)$ be the original codeword and $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ be the distorted codeword, where $e_1, x_1, \tilde{x}_1 \in GF(p^k)$, $e_2, x_2, \tilde{x}_2 \in GF(p^{r_n}), e_3, x_3, \tilde{x}_3 \in GF(p^{r_l})$ and $\tilde{x}_i = x_i \oplus e_i, 1 \leq i \leq 3$. Let $S = f(\tilde{x}_1) \oplus \tilde{x}_2$ be the nonlinear syndrome of the code that can be used for error detection. Assume a standard decoder for V is available. The output of the decoder is composed of the error flag signal E_V and the possible error vector $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$. The values of E_V and the error vector in different situations are defined in Table I.

The detailed error correcting algorithm for code C is described in Algorithm 1. The algorithm only correct errors when information digits are distorted. If errors only occur in the redundant digits or errors are uncorrectable, no correction will be attempted.

Theorem 2: The nonlinear multi-error correcting codes presented in Theorem 1 have no errors miscorrected by all codewords. Any nonzero error will be miscorrected as an error $e = (\hat{e}_1, \hat{e}_2, \hat{e}_3), ||e|| \leq t, \hat{e}_1 \neq 0$ with a probability of at most P_f.

Proof: In Algorithm 1, an error e will be miscorrected if and only if $E_V > 0$, $\hat{e}_1 \neq 0$, $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$ and $\hat{S} = 0$. When $E_V > 0$, after correcting possible errors in $\tilde{x}_1$ and $\tilde{x}_2$, the effective error vector when computing $\hat{S}$ is $(e_1 \oplus \hat{e}_1, e_2 \oplus \hat{e}_2)$. Thereby $\hat{S} = f(x_1 \oplus e_1 \oplus \hat{e}_1) \oplus x_2 \oplus e_2 \oplus \hat{e}_2$. Since $x_2 = f(x_1)$, $\hat{S} = 0$ can be re-written as

$$f(x_1 \oplus e_1 \oplus \hat{e}_1) \oplus f(x_1) \oplus e_2 \oplus \hat{e}_2 = 0. \quad (8)$$

Let H be the parity check matrix of V. When (e_1, e_2, e_3) is miscorrected as $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$, we have $H(\hat{e}_1, \hat{e}_2, \hat{e}_3) = H(e_1, e_2, e_3)$. Thereby $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \oplus (e_1, e_2, e_3)$ is a codeword of V. If $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$, then $(\hat{e}_1, \hat{e}_2)$ can not be equal to (e_1, e_2). Otherwise to guarantee that $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \oplus (e_1, e_2, e_3)$ is a codeword of V, $\hat{e}_3$ has to be equal to e_3, which

TABLE I: The output of the decoder for linear codes that can correct up to t errors

Cases	Error Vector	Error Flag Signal E_V
No errors are detected	$\mathbf{0}$	0
Errors of multiplicity at most t are detected	Correctable error vector	Multiplicity of the error
Errors of multiplicity larger than t are detected	$\mathbf{0}$	-1

Algorithm 1: Error correcting algorithm for the nonlinear multi-error correcting codes in Theorem 1

Input : $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$
Output: $e = (e_1, e_2, e_3)$, ERR

begin
  Decode V, compute S;
  if $E_V = 0, S = 0$ **then**
    No errors are detected, $ERR = 0$;
  else if $E_V = 0, S \neq 0$ **then**
    Uncorrectable multi-errors are detected, $ERR = 1$;
  else if $E_V = -1$ **then**
    Uncorrectable multi-errors are detected, $ERR = 1$;
  else
    $E_V > 0$;
    if $\hat{e}_1 = 0$ **then**
      Errors in the redundant digits are detected, $ERR = 0$;
    else
      Compute $\hat{x}_1 = \tilde{x}_1 \oplus \hat{e}_1, \hat{x}_2 = \tilde{x}_2 \oplus \hat{e}_2$;
      Compute $\hat{S} = f(\hat{x}_1) \oplus \hat{x}_2$;
      if $\hat{S} = 0$ **then**
        $e = (\hat{e}_1, \hat{e}_2, \hat{e}_3)$, $ERR = 0$;
      else
        Uncorrectable multi-errors are detected, $ERR = 1$

contradicts to the assumption that $(\hat{e}_1, \hat{e}_2, \hat{e}_3) \neq (e_1, e_2, e_3)$. Thereby the miscorrected errors can be divided into two cases as stated below.

1) If $\hat{e}_1 \neq e_1$, from the definition of the nonlinear function, there are at most $P_f p^k$ solutions for x_1 in (8). Thereby, the error will be miscorrected with a probability of at most P_f.
2) If $\hat{e}_1 = e_2$, then $\hat{e}_2 \neq e_2$. The error will always be detected since (8) will never be satisfied.

■

The nonlinear multi-error correcting code presented in Theorem 1 has no undetectable errors and no errors miscorrected by all codewords. It has comparable encoding and decoding area overhead to the linear code V (Section VI-C). When $p = 2^m$, the presented code has the same digit-error and burst-error detecting and correcting capabilities as V. We note that the proposed code has errors that are conditionally detected or miscorrected. However, most of these errors are of a high multiplicity and are less dangerous assuming that errors with small multiplicities are more probable. Moreover, these errors will be detected with a probability of at least $1 - P_f$, where P_f is the nonlinearity of f. Thereby, the reliability of architectures protected by the presented nonlinear multi-error correcting codes can be further improved by reducing the nonlinearity for f. (To reduce the nonlinearity of f, it may be necessary to increase the number of redundant digits for V[22].)

Example 1: $(38, 32, 3)$ shortened Hamming codes are widely used for single-bit error correcting in 32-bit systems. It is easy to prove that the number of undetectable errors and the number of errors miscorrected by all codewords for a $(38, 32, 3)$ shortened Hamming code is 2^{32} and $32 \times (2^{32} - 1)$ respectively. Instead of using the $(38, 32, 3)$ shortened Hamming code, we can use a $(39, 32, 3)$ nonlinear single-bit error correcting code generated by Theorem 1. Let $f(y) : GF(2^{32}) \rightarrow GF(2)$ be a quadratic perfect nonlinear function [22] defined by the following equation

$$f(y_1, y_2, \cdots, y_{32}) = y_1 y_2 \oplus y_3 y_4 \oplus \cdots \oplus y_{31} y_{32}, \quad (9)$$

where all the operations are in the binary field. Let V be a $(39, 33, 3)$ shortened Hamming code. According to Theorem 1, the resulting code $\{y, f(y), \phi(z)\}$, where $z = (y, f(y))$ and ϕ is the encoding function for the $(39, 33, 3)$ Hamming code, is a single-bit error correcting code with no undetectable errors and no errors miscorrected by all codewords. Most of the errors can be 100% detected. The worst case error masking or miscorrecting probability is 0.5, which is the nonlinearity of f. This probability can be further reduced by increasing the number of redundant bits for the nonlinear error correcting code and select f to be a quadratic perfect nonlinear function from $GF(2^{32})$ to $GF(2^m)$, where $m > 1$.

While being able to provide an improved protection of systems comparing to linear error correcting codes, the nonlinear multi-error correcting codes generated by Theorem 1 still have the following shortages. First, it always requires more redundant digits than linear codes with the same error correcting capabilities. Second, the error correcting algorithm can not be easily modified to correct errors beyond the error correcting capability t. Obviously, corrections of some errors with multiplicity larger than t can further improve the reliability of the system. Although in the literature there are works discussing the beyond-t error corrections for linear error correcting codes, the modified algorithm usually has much higher area or timing complexity [23].

To improve the code from the above two aspects, we next present a construction of nonlinear partially robust multi-error correcting codes generalized from Vasil'ev constructions [17]. The presented codes may have the same number of redundant digits as Hamming codes and BCH codes. Moreover, the error correcting algorithm for the presented code can also be used to correct some errors with multiplicity larger than t requiring no extra area and timing overhead compared to schemes that

only correct errors with multiplicities up to t.

B. Generalized Vasil'ev Codes

Vasil'ev constructions were first proposed in [17] to generate perfect nonlinear Hamming codes with $n = 2^k - 1$ and $d = 3$. In [15], [24], we generalized Vasil'ev constructions to generate Hamming codes with arbitrary length n and analyzed the error correcting and detecting capabilities of the codes. The presented codes required the same number of redundant digits as linear (shortened) Hamming codes. In this paper, Vasil'ev constructions will be further generalized to construct codes with any given distance d and dimension k.

Theorem 3: Let $q_1 = p^{l_1}$ and $q_2 = p^{l_2}$, $l_1 \geq l_2 \geq 1$. Let V be a (n_1, k_1, d) q_1-ary code and $U = \{(u, uP)\}$ be a (n_2, k_2, d') q_2-ary code, where $u \in GF(q_2^{k_2}), k_2 \leq n_1$, $r_2 = n_2 - k_2$, $d' \geq d - 1$ and P is a $k_2 \times r_2$ encoding matrix in $GF(q_2)$ (the last r_2 columns of the generator matrix of the code in standard form). Let $f : GF(q_1^{k_1}) \rightarrow GF(q_2^{r_2})$ be an arbitrary mapping such that $f(\mathbf{0}), \mathbf{0} \in GF(q_1^{k_1})$ is equal to zero and $f(y) \oplus f(y') \neq f(y \oplus y')$ for some $y, y' \in GF(q_1^{k_1})$, where $\oplus$ is the digit-wise addition in $GF(p)$. Let $u = (u_1, u_2, \cdots u_{k_2})$, where $u_i \in GF(q_2)$. Let $\beta(u) = ((u_1, \mathbf{0}), (u_2, \mathbf{0}), \cdots, (u_{k_2}, \mathbf{0}))$, where $\mathbf{0} \in GF(p^{l_1 - l_2}), (u_i, \mathbf{0}) \in GF(q_1)$ and $\beta(u) \in GF(q_1^{k_2})$. Denote by v_k the information bits of $v \in V$. The code defined by

$$C = \{(u, (\beta(u), \mathbf{0}) \oplus v, uP \oplus f(v_k))\}, \mathbf{0} \in GF(q_1^{n_1 - k_2}) \quad (10)$$

is a $(l_1 n_1 + l_2 n_2, l_1 k_1 + l_2 k_2, d)$ p-ary partially robust code with $|K_d| = p^{l_2 k_2}$. The remaining errors are detected with a probability of at least $1 - P_f$, where P_f is the nonlinearity of f. Consider elements of U in $GF(q_2)$ and elements of V in $GF(q_1)$ as equivalent **digits**. The codeword of C has $n_1 + n_2$ digits and C has the same digit error correcting capability as V.

Proof: Let $c = (u, (\beta(u), \mathbf{0}) \oplus v, uP \oplus f(v_k)), c' = (u', (\beta(u'), \mathbf{0}) \oplus v', u'P \oplus f(v'_k))$ be two codewords of C. The Hamming distance between c and c' is

$$\begin{aligned} ||c \oplus c'|| &= ||u \oplus u'|| + ||(\beta(u), \mathbf{0}) \oplus v \oplus (\beta(u'), \mathbf{0}) \oplus v'|| \\ &+ ||uP \oplus f(v_k) \oplus u'P \oplus f(v'_k)|| \\ &\geq ||v \oplus v'||. \end{aligned}$$

1) If $v \neq v'$, $||c \oplus c'|| \geq d$ because the Hamming distance of V is d.
2) If $v = v'$, $||c \oplus c'|| = 2 \times ||u \oplus u'|| + ||uP \oplus uP'|| \geq d$ because the Hamming distance of $U = \{(u, uP)\}$ is at least $d - 1$.

Thereby, the Hamming distance of C is d. We say that an error e is masked by a codeword c if $e \oplus c = c' \in C$. Let H be the parity check matrix of V. An error $e = (e_1, e_2, e_3)$ where $e_1 \in GF(q_2^{k_2}), e_2 \in GF(q_1^{n_1})$ and $e_3 \in GF(q_2^{r_2})$ is masked if and only if $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is zero and $f(\tilde{v}_k) \oplus f(v_k) \oplus e_1 P \oplus e_3 = 0$, where $\tilde{v}_k$ is the information part of $\tilde{v} = v \oplus (\beta(e_1), \mathbf{0}) \oplus e_2$ and $\mathbf{0} \in GF(q_1^{n_1 - k_2})$. The errors can be divided into four classes as follows.

1) $(\beta(e_1), \mathbf{0}) = e_2$ and $e_1 P = e_3$. The error will always be masked. The number of errors in this class is $q_2^{k_2}$;
2) $(\beta(e_1), \mathbf{0}) = e_2$ but $e_1 P \neq e_3$. The error will always be detected. There are $q_2^{n_2} - q_2^{k_2}$ errors belonging to this class;
3) $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is zero but $(\beta(e)_1, \mathbf{0}) \neq e_2$. The error masking probability depends on the nonlinear function f. In the worst case, a specific error will be masked by $P_f \times |C|$ codewords. The number of errors in this class is $q_2^{n_2}(q_1^{k_1} - 1)$;
4) $H((\beta(e_1), \mathbf{0}) \oplus e_2)$ is not zero. The error will always be detected. The number of errors is $q_2^{n_2}(q_1^{n_1} - q_1^{k_1})$.

■

Remark 1: Binary Vasil'ev code presented in [17] is a special case where $q_1 = q_2 = 2$, $\{(u, Pu)\}$ is a 1-d parity code with minimum distance two and V is a perfect Hamming code.

Some nonlinear multi-error correcting codes as good as linear codes in terms of the number of redundant digits for the same distance and length can be generated based on the above construction.

Example 2: In [25], it was shown that the largest possible k for binary codes with $n = 63$ and $d = 5$ is 52. Let V be a $(63, 52, 5)$ binary code ($q_1 = 2$). Let $\{(u, uP)\}$ be a $(4, 1, 4)$ binary repetition code that contains only 2 codewords 0000 and 1111 ($q_2 = 2$). Select f to be a quadratic perfect nonlinear function $f = s_1 \bullet s_2 \oplus s_3 \bullet s_4 \oplus \cdots \oplus s_{13} \bullet s_{14}$ with $P_f = \frac{1}{8}$ where $s_i \in GF(2^3)$ and $\bullet$ is the multiplication in $GF(2^3)$. A $(67, 53, 5)$ partially robust double-bit error correcting code can be constructed as described in Theorem 3. This code has Hamming distance five and only one undetectable nonzero error with the same number of redundant bits as BCH codes. All the other errors are detectable with a probability of at least 0.875.

Theorem 3 can also generate nonbinary multi-error correcting codes that are as good as linear codes in terms of the number of redundant digits.

Example 3: In general, a nonbinary BCH code with Hamming distance d in $GF(q)$ has length $n = q^m - 1$ and dimension $k = q^m - 1 - (d - 1)m$. Let $n = 49$, $q = 7$ and $d = 5$. The corresponding nonbinary BCH code is constructed by shortening the BCH code with $m = 3$ and $k = 7^3 - 13$. The number of redundant digits is 12. Let V be a $(48, 40, 5)$ nonbinary BCH code in $GF(7)$ ($m = 2$), U be a $(5, 1, 5)$ repetition code in $GF(7)$ and f be a perfect nonlinear function from $GF(7^{40})$ to $GF(7^4)$. The resulting nonlinear nonbinary BCH code constructed as in Theorem 3 has the same number of redundant digits as the BCH codes in $GF(7)$. Only 7 errors are undetectable (including the all-zero error vector). All other errors are detected with a probability of at least $1 - 7^{-4}$.

We next describe the error correcting algorithm for the nonlinear multi-error correcting codes presented in Theorem 3. We consider elements of U in $GF(q_2)$ and elements of V in $GF(q_1)$ as equivalent digits. The multiplicity of the error is defined as the number of nonzero digits in the error vector. Let $e = (e_1, e_2, e_3)$ be the error vector and $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$ be the distorted codeword, where $e_1, \tilde{x}_1 \in GF(q_2^{k_2})$, $e_2, \tilde{x}_2 \in GF(q_1^{n_1}), e_3, \tilde{x}_3 \in GF(q_2^{r_2})$ and $\tilde{x}_i = x_i \oplus e_i, 1 \leq i \leq 3$. Denote by $\tilde{v} = (\beta(\tilde{x}_1), \mathbf{0}) \oplus \tilde{x}_2$ the distorted codeword in

V and $\tilde{v}_k$ the information part of $\tilde{v}$. In the presented error correcting algorithm, we assume that $d' = d$ (see Theorem 3) and the standard error correcting algorithms are available for V and U. After receiving the possibly distorted codeword $(\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$, compute $S = \tilde{x}_1 P \oplus f(\tilde{v}_k) \oplus \tilde{x}_3$. Decode $\tilde{v}$ using the standard error correcting algorithm for V. The output of the decoder for V contains two parts. One is the decoded error vector $\hat{e}_2$ in $GF(q_1^{n_1})$ and the other is the error flag signal E_V. Similarly, the outputs of the decoder for U contain the error flag signal E_U and the possible error patterns for the first and the third part of the codeword, which are $\hat{e}_1 \in GF(q_2^{k_2})$ and $\hat{e}_3 \in GF(q_2^{r_2})$ respectively. The values of the output signals of the decoders for U and V in different cases are described in Table I. The detailed error correcting algorithm for the nonlinear multi-error correcting code presented in Theorem 3 is shown in Algorithm 2.

In reality, the error will be corrected only if at least one of the information bits is distorted. If all errors are in the redundant bits, no correction will be attempted. The miscorrection of errors for C is strongly related to the error correcting properties of U and V. To simplify the analysis, we assume that U and V are both linear codes. For a given $e, 1 \leq ||e|| \leq t$, all vectors belonging to the coset in which e is the coset leader will be miscorrected as e by the linear code. For any $(n, k, d = 2t + 1)$ binary linear code that can correct up to t errors, for example, the number of miscorrected errors is $\sum_{i=1}^{t} \binom{n}{i}(2^k - 1)$. Compared to linear codes, the number of miscorrected errors for the proposed nonlinear multi-error correcting codes will be drastically reduced as shown in the following Theorem.

Theorem 4: Assume $d' = d$ in Theorem 3 and f is a perfect nonlinear function from $GF(q_1^{k_1})$ to $GF(q_2^{r_2})$. Suppose only errors occurring to the information bits are corrected, denote by T the number of correctable error patterns by the nonlinear multi-error correcting codes presented in Theorem 3. The number of errors miscorrected by all codewords of the multi-error correcting codes presented in Theorem 3 is $|K_c| = (q_2^{k_2} - 1)T$.

Proof: According to Algorithm 2, the occurrence of miscorrections can be divided into the following cases.

1) When $d > 3$, $E_V = 0, S \neq 0$ and $E_U > 0$, the error pattern is $e = (\hat{e}_1, (\beta(\hat{e}_1), 0), \hat{e}_3)$ which is generated by the decoder of U. Errors are miscorrected by C if and only if they are miscorrected by the linear code U. Since e will be corrected only if $\hat{e}_1 \neq 0$ and $||e|| \leq t$, the number of possible correctable error patterns is

$$\sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} \sum_{j=0}^{t-2i} \binom{k_2}{i} \binom{r_2}{j} (q_2 - 1)^{i+j}. \quad (11)$$

Thereby the number of miscorrected errors in this class is

$$(q_2^{k_2} - 1) \times \sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} \sum_{j=0}^{t-2i} \binom{k_2}{i} \binom{r_2}{j} (q_2 - 1)^{i+j}. \quad (12)$$

2) When $E_V > 0$,

$$\hat{x}_3 = \tilde{x}_3 \oplus f(\hat{v}_k) = x_1 P \oplus f(v_k) \oplus f(\hat{v}_k) \oplus e_3. \quad (13)$$

Algorithm 2: Error correcting algorithm for the nonlinear multi-error correcting codes in Theorem 3

Input : $\tilde{c} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)$
Output: $e = (e_1, e_2, e_3)$, ERR

begin
  Decode V, compute $\hat{e}_2, E_V, S$;
  if $E_V = 0, S = 0$ **then**
    No errors are detected, $ERR = 0$;
  else if $E_V = 0, S \neq 0$ **then**
    if $d = 3$ **then**
      Errors either occur only in the redundant bits or are uncorrectable multi-errors;
      $ERR = 1$;
    else
      Decode U, compute $\hat{e}_1, \hat{e}_3, E_U$;
      if $E_U > 0$ **then**
        $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}), \hat{e}_3)$;
        e will only be corrected when $||e|| \leq t$;
        $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
      else
        Uncorrectable muti-errors are detected, $ERR = 1$;
  else if $E_V = -1$ **then**
    Uncorrectable muti-errors are detected, $ERR = 1$;
  else
    E_V is larger than 0;
    Compute: ;
      $\hat{x}_2 = \tilde{x}_2 \oplus \hat{e}_2$;
      $\hat{v} = (\beta(\tilde{x}_1), \mathbf{0}) \oplus \hat{x}_2$;
      $\hat{x}_3 = \tilde{x}_3 \oplus f(\hat{v}_k)$ ($\hat{v}_k$ is the information part of $\hat{v}$);
    Decode U according to $\tilde{x}_1$ and $\hat{x}_3$;
    if $E_U = 0$ **then**
      $e = (0, \hat{e}_2, 0)$;
      e will be only corrected if there are errors in the information bits;
      $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
    else if $E_U > 0$ **then**
      $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}) \oplus \hat{e}_2, \hat{e}_3)$;
      e will only be corrected when $||e|| \leq t$ and there are errors in the information bits;
      $ERR = 0$ when errors are corrected and $ERR = 1$ otherwise;
    else
      $E_U = -1$, uncorrectable multi-errors are detected, $ERR = 1$.

After correcting $\hat{e}_2$, the error pattern visible to U is $(e_1, f(v_k) \oplus f(\hat{v}_k) \oplus e_3)$.

a) If $\hat{v}_k \neq v_k$ ($(e_1, \mathbf{0}) \oplus e_2$ is miscorrected by V), $f(v_k) \oplus f(\hat{v}_k) \oplus e_3$ may vary for different information bits. An error $(e_1, f(v_k) \oplus f(\hat{v}_k) \oplus e_3)$ will be miscorrected if and only if it belongs to the same coset as correctable errors (including the all-zero error vector) by U. Since errors are only corrected when $||e|| \leq t$, it is easy to verify that none of errors in this class will always be miscorrected.

b) If $\hat{v}_k = v_k$ ($(e_1, \mathbf{0}) \oplus e_2$ is successfully corrected by V), the error pattern visible to U will be (e_1, e_3). Errors in this class are miscorrected if and only if (e_1, e_3) is miscorrected by U.

i) When $E_U \geq 0$ and $\hat{e}_1 = 0$, the error will be only corrected if $\hat{e}_2$ has errors in the information bits and $||e|| \leq t$. The number of correctable error patterns is given by (14). The number of miscorrected errors in this situation is given by (15).

ii) When $E_U > 0$ and $\hat{e}_1 \neq 0$, the number of correctable error patterns is (16). The number of miscorrected errors in this situation is given by (17)

■

Nonlinear multi-error correcting codes presented in Theorem 3 still have undetectable errors and errors miscorrected by all codewords. However, the number of these errors are drastically reduced compared to linear codes with the same length and the same error correcting capability (see Section V).

In Theorem 3, elements of U and V can belong to different fields, which allows a more flexible selection of the two codes. For example, in order to construct a nonlinear 5-digit error correcting code with elements in $GF(2^{10})$, we can select V to be a linear 5-digit error correcting Reed-Solomon code in $GF(2^{10})$ and U to be a binary repetition code with $n = 11$ and $k = 1$ with elements in $GF(2)$. The resulting code will have the same digit-error correcting capabilities, much less undetectable and miscorrected errors at the cost of only one more redundant digit in $GF(2^{10})$ (redundant bits of U) when comparing to V.

The codes presented in Theorem 3 may be as good as BCH codes in terms of the number of redundant digits (Example 2). Moreover, Algorithm 2 can be slightly modified to correct errors with multiplicities larger than t. For example, when $E_V = 0, S \neq 0$ and $E_U > 0$, the potential error pattern is $e = (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}), \hat{e}_3)$. The original algorithm only correct errors when $||e|| \leq t$. This requirement can be removed so that some errors with multiplicity larger than t can also be corrected.

Example 4: In this example we describe the encoding and decoding procedure of a $(31, 17, 5)$ binary nonlinear 2-error-correcting code. Let V be a $(26, 16, 5)$ BCH code whose generator polynomial is $g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1$. Let $U = \{(u, uP)\}$ be a repetition code, where $u \in GF(2), uP \in GF(2^4)$. Select f to be a quadratic perfect nonlinear function from $GF(2^{16})$ to $GF(2^4)$ defined by $f = s_1 \bullet s_2 \oplus s_3 \bullet s_4$, where $\oplus$ is the bit-wise XOR and $\bullet$ is the multiplication in $GF(2^4)$. Let (10101100111101001) be the 17-bit message that needs to be encoded. Then $u = 1$, $v_k = (1101100111101001)$. The redundant bits for V is (0101110001) and $Pu \oplus f(v_k) = (1001)$. Thereby the entire codeword is

$$c = (10101100111101001010111000110 01).$$

Suppose the four left-most bits are distorted. The distorted codeword is

$$\tilde{c} = (01011100111101001010111000110 01).$$

Thereby we have

$$\tilde{v} = (10111001111010010101110001).$$

The decoder will correct the 2-bit error $\hat{e}_2 = (01100000000000000000000000)$ in $\tilde{v}$. After this, $\hat{v}$ and $\hat{x}_3$ are re-computed and U is decoded according to $\tilde{x}_1$ and $\hat{x}_3$. It is easy to verify that $\hat{x}_3 = (1111)$. Since $\tilde{x}_1 = 0$ (the first bit of $\tilde{c}$), the input to the decoder of U is (01111). An error in the first bit will be successfully corrected by U. Hence we have

$$\begin{aligned} e &= (\hat{e}_1, (\beta(\hat{e}_1), \mathbf{0}) \oplus \hat{e}_2, \hat{e}_3) \\ &= (11110000000000000000000000000). \end{aligned}$$

Thereby, an error of multiplicity four is successfully corrected although the Hamming distance of the code is only five.

We note that correcting errors with multiplicity larger than t will result in more errors that are miscorrected by all codewords. For example, when $E_V = 0, S = 0$ and $E_U > 0$, the error $e = (e_1, e_2, e_3)$ is miscorrected if and only if (e_1, e_3) is miscorrected by U. Suppose no correction will be attempted if $||e|| > t$, errors will not be corrected if $||\hat{e}_1|| + ||\hat{e}_3|| = t$ since in that case the resulting error vector $(\hat{e}_1, (\beta(\hat{e}_1), (0)), \hat{e}_3)$ will have a multiplicity larger than t. As a result, no miscorrections will happen when $(\hat{e}_1, \hat{e}_3) \neq (e_1, e_3)$, $||\hat{e}_1|| + ||\hat{e}_3|| = t$. Without the limitation on $||e||$, miscorrections will occur when (e_1, e_3) with a multiplicity of $t+1$ is mistakenly corrected as $(\hat{e}_1, \hat{e}_3)$ whose multiplicity is t. Thereby, there is a tradeoff between the number of correctable error patterns and the size of the correction kernel $|K_C|$. The decision of whether and how to modify Algorithm 2 should be made according to specific applications and the estimated error models.

V. Alternatives for the Protection of MLC NAND Flash Memories

As case studies, in this section we compare six binary 5-bit error correcting codes for the protection of MLC NAND flash memories with 1024 data bytes in each page. For MLC NAND flash memories with a larger page size, longer codes generated by Theorem 1 and 3 can be used and all the analysis and comparison can be easily adjusted to justify the advantage of the presented codes.

The first two alternatives are the widely used $(8262, 8192, 11)$ BCH code [10] and the $(830, 820, 11)$ shortened Reed-Solomon code defined over $GF(2^{10})$ [12].

$$\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_1}{i}\binom{r_1}{j}\binom{r_2}{l}(q_1-1)^{i+j}(q_2-1)^{l}. \tag{14}$$

$$(q_2^{k_2}-1)\times\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_1}{i}\binom{r_1}{j}\binom{r_2}{l}(q_1-1)^{i+j}(q_2-1)^{l}. \tag{15}$$

$$\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_2}{i}\binom{n_1}{j}\binom{r_2}{l}(q_1-1)^{j}(q_2-1)^{i+l}-\sum_{i=1}^{\lfloor t/2\rfloor}\sum_{j=0}^{t-2i}\binom{k_2}{i}\binom{r_2}{j}(q_2-1)^{i+j}. \tag{16}$$

$$(q_2^{k_2}-1)\times(\sum_{i=1}^{t}\sum_{j=0}^{t-i}\sum_{l=0}^{t-i-j}\binom{k_2}{i}\binom{n_1}{j}\binom{r_2}{l}(q_1-1)^{j}(q_2-1)^{i+l}-\sum_{i=1}^{\lfloor t/2\rfloor}\sum_{j=0}^{t-2i}\binom{k_2}{i}\binom{r_2}{j}(q_2-1)^{i+j}). \tag{17}$$

The third and the forth alternatives are based on Theorem 1. Let $p=2, k=8200, r_n=10$. Select f to be a quadratic perfect nonlinear function defined by the following equation.

$$f(y)=y_1\bullet y_2\oplus y_3\bullet y_4\cdots y_{819}\bullet y_{820}, \tag{18}$$

where $y_i\in GF(2^{10}), 1\le i\le 820$ and $\bullet$ is the multiplication in $GF(2^{10})$. Let V be a $(8280, 8210, 11)$ BCH code. The codeword of the resulting nonlinear multi-bit correcting code constructed as described in Theorem 1 is in the format of $(y, f(y), \phi(z))$, where $z=(y, f(y))$, $y\in GF(2^{8200})$ are the information bits, $f(y)\in GF(2^{10})$ are the nonlinear redundant bits and $\phi(z)\in GF(2^{70})$ are the linear redundant bits. The code is a 5-bit error correcting code with length 8280 and dimension 8200.

Let $p=2^{10}, k=820$ and $r_n=1$. Let f be the same quadratic function defined in (18) and V be a $(831, 821, 11)$ shortened Reed-Solomon codes in $GF(2^{10})$. A $(831, 820, 11)$ nonlinear multi-digit error correcting code can be constructed by Theorem 1. The code has the same bit-error correcting capabilities as the $(8262, 8192, 11)$ BCH codes. Moreover, the code can also correct burst errors and up to 5-digit errors like $(830, 820, 11)$ Reed-Solomon codes defined over $GF(2^{10})$.

The fifth and the sixth alternatives are based on Theorem 3. For these two alternatives, f is still selected to be the quadratic function defined in (18). Let $q_1=q_2=2$. Let V be a $(8270, 8200, 11)$ BCH code and U be a $(11, 1, 11)$ linear repetition code. The nonlinear code constructed as described in Theorem 3 is a $(8281, 8200, 11)$ nonlinear 5-bit error correcting code. The code can also correct some errors with multiplicities higher than 5 without any extra overhead.

Let $q_1=2^{10}$ and $q_2=2$ in Theorem 3. Let V be a $(830, 820, 11)$ shortened Reed-Solomon code defined over $GF(2^{10})$ and U be a $(11, 1, 11)$ repetition code in binary field. The resulting code in Theorem 3 is a $(8301, 8201, 11)$ nonlinear 5-digit error correcting code.(The length and the dimension here are in terms of the number of binary bits.) The code can correct up to 5-digit errors in $GF(2^{10})$ and has the same burst error correcting capabilities as Reed-Solomon codes. (The first bit is treated as a separate digit.) Like alternative five, the nonlinear multi-digit error correcting code based on Theorem 3 can also correct some errors with more than 5 digits distorted.

Table II compares the error correcting properties of the six 5-bit error correcting codes. BCH code and Reed-Solomon code have a large number of undetectable errors. Let us denote the number of codewords of multiplicity i by A_i. For every codeword c of multiplicity eleven belonging to the BCH code, if $e\oplus e'=c, ||e||=5$ and $||e'||=6$, $||e'||$ will be miscorrected by all codewords as e since they have the same syndrome. Thereby, the number of errors of multiplicity six miscorrected by all codewords of the BCH code is $A_{11}\binom{11}{6}$. (If the error is only corrected when there is at least one information bit distorted, this number will be a little smaller.) According to the results presented in [25], for a $(8262, 8192, 11)$ shortened BCH code, A_{11} can be roughly estimated by $A_{11}=\binom{8262}{11}/2^{70}\approx 10^{14}$. Thereby, a large number of errors with multiplicity 6 will be miscorrected by the BCH codes. The fraction F of errors that are always miscorrected by the BCH code can be calculated as

$$F=\frac{(2^k-1)\sum_{i=1}^{t}\binom{n}{i}}{2^n}\approx\frac{\sum_{i=1}^{t}\binom{n}{i}}{2^{n-k}}. \tag{19}$$

For the linear $(8262, 8192, 11)$ BCH code, $F\approx 10^{-4}$.

For the Reed-Solomon code, a 6-bit error is miscorrected if and only if (1) the 6 bits spread over 6 digits in $GF(2^{10})$; and (2) there is a codeword with 11 digits that contains the 6 digits in (1). Moreover, given the fact that the length of the Reed-Solomon code (in terms of digits) is much smaller than the length of the BCH code (in terms of bits), the number of miscorrected 6-bit errors for Reed-Solomon codes will be much smaller than that for BCH codes. However, similar to (19), we can compute the proportion of errors that are always miscorrected by the Reed-Solomon code, which is of the order of 10^{-3} and is worse than the BCH code.

Codes 3 and 4 generated by Theorem 1 do not have undetectable and miscorrected errors. Codes 5 and 6 based on Theorem 3 have only 1 undetectable error. When no error with multiplicity larger than 5 is corrected, according to the proof of Theorem 3 it is easy to derive that the smallest possible multiplicity of errors that are always miscorrected by code 5 or 6 is seven. Moreover, since $q_2=2, k_2=1$, the number of errors that are miscorrected by all codewords is equal to the number of correctable errors. As a result, the fraction of errors that are always miscorrected by these two codes is almost 0. When each non-zero error pattern is equi-probable,

TABLE II: Comparison of six 5-bit error correcting codes for the protection of MLC NAND flash memories

	BCH	Reed-Solomon$^{(a)}$	Code 3 (Theo 1)	Code 4 (Theo 1)	Code 5 (Theo 3)	Code 6 (Theo 3)
$n^{(b)}$	8262	8300	8280	8310	8281	8301
$k^{(b)}$	8192	8200	8200	8200	8201	8201
Digit & Burst Error Correcting	No	Yes	No	Yes	No	Yes
Beyond 5-bit Error Correcting	No	No	No	No	Yes	Yes
Number of Undetectable Errors $^{(e)}$	2^{8192}	2^{8200}	0	0	1	1
Miscorrected Errors $^{(e)}$, $\|\|e\|\| = 6$	$\approx 10^{17}$	$< A_{11}^{*}\binom{11}{6}^{(c)}$	0	0	$0^{(d)}$	$0^{(d)}$
Fraction of Miscorrected Errors	$\approx 10^{-4}$	$\approx 10^{-3}$	0	0	≈ 0	≈ 0

(a): The code is defined in $GF(2^{10})$.
(b): The length and the dimension here are in terms of the number of bits in binary field.
(c): A_{11}^{*} is the number of codewords in a code over $GF(2^{10})$ with 11 nonzero digits.
(d): The number is for the case when no errors with multiplicity larger than 5 are corrected.
(e): Errors undetectable or miscorrected by all codewords. Besides these two types of errors, the presented codes also have errors that are conditionally detectable or conditionally miscorrected, most of which can be detected with a probability of $1 - 2^{-10}$.

the probability of miscorrection for the presented nonlinear multi-error correcting codes is much smaller than that for BCH codes and Reed-Solomon codes. Moreover, the presented codes have no errors of multiplicity 6 that are miscorrected by all codewords. Thereby, when assuming a fixed bit error rate, the probability that an error is always miscorrected by these codes is still much smaller than their linear alternatives.

Different from linear codes, the four presented nonlinear multi-error correcting codes have errors that are conditionally undetectable or miscorrected. These errors usually have high multiplicities. Moreover, most of these errors can be detected with a probability $1 - 2^{-10}$. Thereby, the existence of conditionally undetectable or miscorrected errors will not compromise the reliability of NAND flash memories protected by the proposed nonlinear multi-error correcting codes.

We note that the above message-dependent error detecting characteristic of the proposed nonlinear multi-error correcting code is also very helpful for MLC NAND flash memories for the protection of hardware malfunctions such as data retention and programming/erasing endurance failure [4]. Due to the decreased programming voltage margin, data retention is more likely to happen for MLC technology than for SLC technology. The problem of programming/erasing endurance also becomes more serious for MLC NAND flash memories, for which the typical number of supported program/erase cycles is fewer than 10000 [2]. Errors introduced by these hardware failures will never disappear or will only disappear after the next erasing or programming operation. Hence, the proposed nonlinear t-error-correcting code with stronger error detecting and correcting capability for repeating errors due to the message-dependent error detecting characteristic can be used together with other protection schemes to efficiently detect these failures and protect the devices against them.

Generally speaking, the proposed nonlinear multi-error correcting codes require more redundant bits than their linear alternatives. However, this results in only a very small decrease in the code rate (10 more redundant bits are required for more more than 8000 information bits). Moreover, due to the existence of spare bytes in MLC NAND flash memory (Section II), the number of redundant bits of the error correcting codes used for MLC NAND flash memories is not as critical as for other types of memories such as SRAM and DRAM where the area overhead is mostly determined by the number of redundant bits.

VI. Hardware Design of the Encoder and the Decoder for Nonlinear Multi-Error Correcting Codes

In this section, we present the encoder and the decoder architectures for the proposed nonlinear multi-error correcting codes. We estimate the area, the performance and the power consumption of the proposed architectures and compare them to architectures based on BCH codes and Reed-Solomon codes (see Section V).

A. Encoder Architecture

The encoder for BCH codes and Reed-Solomon codes are conventionally implemented based on a linear feedback shift register (LFSR) architecture. Both the serial and the parallel structures for LFSRs are well studied in the community. In general, the serial LFSR needs k clock cycles while the parallel LFSR needs only $\lceil k/q \rceil$ clock cycles to finish the computation of the redundant bits at the cost of higher hardware complexity, where k is the number of information bits and q is the parallelism level of the LFSRs.

Compared to the encoder for the BCH codes and Reed-Solomon codes, the encoder for the proposed nonlinear multi-error correcting codes mainly requires one more finite field multiplier and two registers for the computation of the nonlinear redundant bits. The detailed architecture of the encoder for the nonlinear $(8281, 8201, 11)$ 5-bit error correcting code generated by Theorem 3 is shown in Figure 3. The design is based on the parallel LFSR proposed in [26]. The parallelism level of the design is 10. During each clock cycle, 10 information bits are inputted to the encoder. The most significant bit (msb) of the message is input via a separate port. The first information bit for the BCH code is derived by XORing msb with the first bit of msg at the first clock cycle (when $cnt = 0$ as shown in the figure). The bottom half of the architecture is a parallel LFSR used to generate the redundant bits for BCH codes. D is a 10×70 binary matrix [26]. During each clock cycle, the 10 most significant bits in the shift register are XORed with the new input and then multiplied by D. The output of the multiplier is XORed with the shifted data from the shift

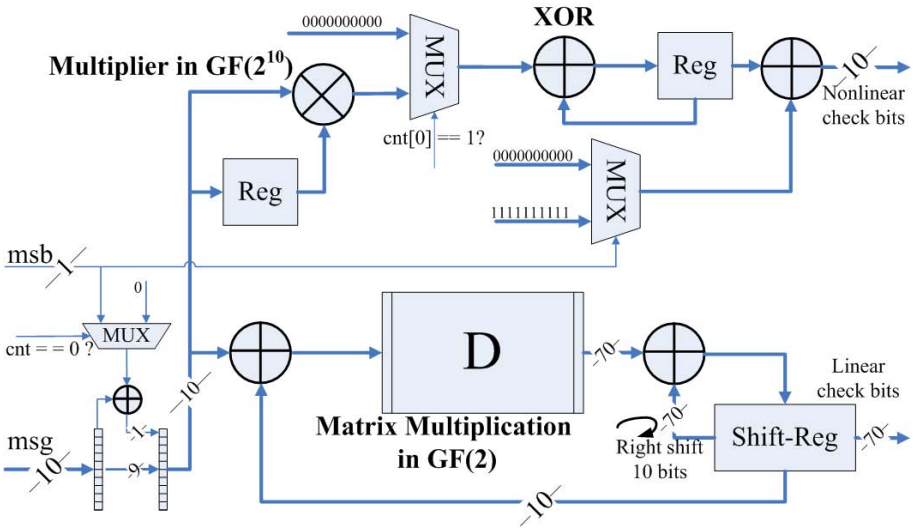


Fig. 3: The architecture of the encoder for the $(8281, 8201, 11)$ nonlinear 5-error-correcting code

register to generate the input to the register. The top half of the architecture is for the computation of nonlinear redundant bits. During the even-numbered clock cycles, the 10-bit input is buffered. During the odd-numbered clock cycles, the buffered data is multiplied by the new input in $GF(2^{10})$ and then added to the output registers. A 10-bit mask is XORed with the data in the output register to generate the nonlinear redundant bits. For the $(8281, 8201, 11)$ 5-error-correcting code, 820 clock cycles are required to complete the encoding of the message.

The encoder for the $(8280, 8200, 11)$ nonlinear 5-bit error correcting code based on Theorem 1 is similar to the one shown in Figure 3. The same structure (top half) is used to compute the 10-bit nonlinear redundant bits. The main difference between the two encoders is as follows. First, the encoder for the $(8280, 8200, 11)$ code does not require a separate port for msb. All information bits are input via msg in 820 clock cycles, assuming a parallelism level of 10. Second, the encoding of the $(8280, 8200, 11)$ code needs one more clock cycle to complete compared to the $(8281, 8201, 11)$ code. At the 821^{th} clock cycle, the input to D (Figure 3) is switched to the already-generated nonlinear check bits using a 10-bit 2:1 multiplexer.

Instead of using a parallel architecture described above, the encoders for the $(830, 820, 11)$ linear shortened Reed-Solomon code defined over $GF(2^{10})$ and the two nonlinear multi-digit error correcting codes presented in Section V can be based on a much simpler serial LFSR. Since the length of the Reed-Solomon code and the nonlinear multi-digit error correcting codes is 10 times shorter than that of the bit-error correcting codes, the number of clock cycles required to complete the encoding for the Reed-Solomon code and the nonlinear error correcting codes will still be the same as for the bit-error correcting codes even with a serial LFSR. The former, however, requires that all operations are performed in $GF(2^{10})$.

B. Decoder Architecture

The decoding of the proposed nonlinear multi-error correcting codes requires the decoding of a BCH code or a Reed-Solomon code. The standard decoder for the BCH codes mainly contains three parts: the syndrome computation block, the error locator polynomial generation block and the Chien search block [27]. Compared to the decoder for the BCH codes, the decoder for the Reed-Solomon codes requires one more block to compute the error magnitude. We next briefly discuss the implementation of the above four blocks and then present the decoder architecture for the proposed nonlinear multi-error correcting codes.

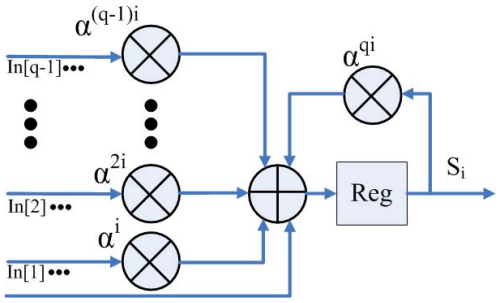


Fig. 4: The syndrome computation block with a parallelism level of q for BCH codes

1) Syndrome Computation: Without loss of generality, assume that the BCH code is a narrow-sense BCH code [25]. Let us denote the received codeword by $\tilde{c} = (\tilde{x}_1, \tilde{x}_2 \cdots \tilde{x}_{n-1}, \tilde{x}_n)$. For a $(n, k, d = 2t + 1)$ t-error-correcting BCH codes, the syndromes are defined as $S_i = \sum_{j=0}^{n-1} x_{j+1}\alpha^{ij}, 0 \leq i \leq 2t-1$, where α is the primitive element of $GF(2^m)$. For binary BCH codes, $S_{2i} = S_i^2$. Hence only odd-numbered S_i needs to be computed from $\tilde{c}$. The other syndromes can be computed using a much simpler square circuit in $GF(2^m)$. To improve the throughput of the decoder, a parallel design can be applied to process multiple bits per clock cycle. Figure 4 shows the syndrome computation circuit with a parallelism level of q for

one S_i. For the whole syndrome computation block, t such structures are needed.

2) Error Locator Polynomial Generation: After the syndromes are computed, the error locator polynomial Λ will be generated using the Berlekamp-Massey (BM) algorithm. The hardware implementations of the BM algorithms have been well studied in the community [28], [29], [30], [31]. In our design a fully serial structure proposed in [28] is used to minimize the area overhead. The design mainly requires three multipliers in $GF(2^m)$ and two FIFOs. The error locator polynomial Λ of degree t can be generated in $t(t+3)/2$ clock cycles. For our design, $t = 5$ and 20 clock cycles are needed for the generation of Λ.

3) Chien Search: Let us denote the primitive element in $GF(2^m)$ by α. The Chien search algorithm exhaustively tests whether α^i is a root of the error locator polynomial Λ. If $\Lambda(\alpha^i) = 0$, the error location is $2^m - 1 - i$. Rewrite $\Lambda(\alpha^i)$ as:

$$\begin{aligned}\Lambda(\alpha^i) &= \lambda_0 \oplus \lambda_1\alpha^i \oplus \lambda_2\alpha^{2i} \oplus \cdots \oplus \lambda_t\alpha^{ti} \\ &= \lambda_{0,i} \oplus \lambda_{1,i}\alpha \oplus \lambda_{2,i}\alpha^2 \oplus \cdots \oplus \lambda_{t,i}\alpha^t. \quad (20)\end{aligned}$$

The computation complexity is reduced based on the fact that $\lambda_{j,i+1} = \lambda_{j,i}\alpha^j, 0 \le j \le t$. The algorithm can also be paralleled to test multiple positions per clock cycle. A typical implementation of the algorithm with a parallelism level of q contains t m-bit multiplexers and registers, $q \times t$ multipliers for multiplication by a constant and q adders in $GF(2^m)$ [32]. In [27], a strength-reduced parallel Chien search architecture is proposed. The authors showed that by a simple transformation of the error locator polynomial, most of the Galois field multiplications can be replaced by shift operations resulting in much lower hardware complexity (Figure 5). For the detail of the architecture, please refer to [27].

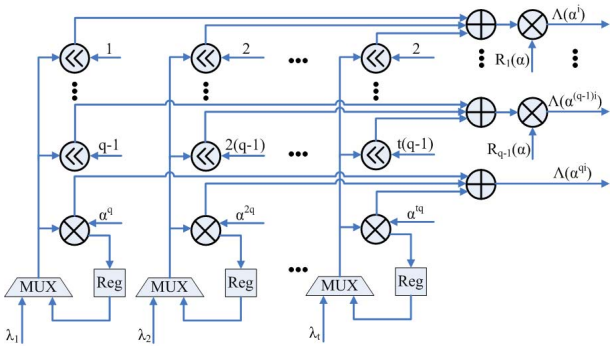


Fig. 5: Strength-reduced Chien search architecture with a parallelism level of q

4) Error Magnitude Computation for Reed-Solomon Codes: Besides the error locator polynomial Λ, the Berlekamp-Massey algorithm can also generate the error magnitude polynomial $\Omega(z)$ defined by

$$(1 + S(z))\Lambda(z) = \Omega(z) \; mod \; z^{2t}, \quad (21)$$

where $S(z) = S_0 \oplus S_1 z \oplus \cdots \oplus S_{2t-1}z^{2t-1}$ is the syndrome polynomial. According to Forney's algorithm [33], the error magnitude at position i can be computed as

$$e_i = \frac{z^b\Omega(z)}{z\Lambda'(z)}, z = \alpha^{-i}, \quad (22)$$

where $\Lambda'(z)$ is the derivative of $\Lambda(z)$ and b is an integer. It is easy to verify that $\Lambda'(z)$ is simply the sum of the terms with odd degrees in $\Lambda(z)$ and can be directly derived during the computation of Λ.

5) Decoder Architecture for the Nonlinear Multi-Error Correcting Codes: The decoder for the nonlinear multi-error correcting codes presented in Theorem 1 is similar to the decoders for BCH codes and Reed-Solomon codes. In fact, most of the decoding can be completed by the standard BCH or Reed-Solomon decoder. The main difference is as follows. First, the nonlinear multi-error correcting codes need to compute the nonlinear syndrome S (see Algorithm 1) when receiving the possibly distorted codewords and re-compute S after correcting errors located by V. Second, after the decoding of the linear codes is completed and S is re-computed, one more clock cycle is required for the decoder of the nonlinear code to verify the error correcting results so that possible miscorrection of errors can be prevented.

The decoder for the nonlinear multi-error correcting codes based on Theorem 3 is slightly more complicated than the decoder for codes based on Theorem 1. As an example, the detailed architecture of the decoder for the $(8281, 8201, 11)$ nonlinear 5-bit error correcting code is shown in Figure 6. The whole decoding procedure requires 1675 clock cycles assuming a parallelism level of 10. During the first 827 cycles, S and the syndrome of the BCH code are computed. If no errors are detected by the BCH code, the decoding procedure will be completed at the 828^{th} clock cycle. Depending on the value of S, either the first two information bits will be flipped or ERR will be pulled down by the ERR generating circuit which indicates that there are no errors occurring to the information bits of the code. The the error locator polynomial generation and the Chien search will be incurred only when errors are detected by the BCH code, which can effectively reduce the average decoding latency.

If errors are detected by the BCH code, the Berlekamp-Massey algorithm will take another 20 clock cycles to generate the error locator polynomial Λ. After this the Chien search block will exhaustively test all possible error locations. If $\Lambda(\alpha^i) = 0$, then the error location is $2^m - 1 - i$. Since a $(8270, 8200, 11)$ shortened BCH code is used, only $\Lambda(\alpha^i), 8114 \le i \le 16383$ $(m = 14)$ need to be computed. The original strength-reduced Chien search architecture is slightly modified for the decoding of shortened BCH codes. The constant inputs $\lambda_i, 1 \le i \le t$ to the bottom t Galois field multipliers in Figure 5 are set to be α^{-10i} instead of α^{10i} $(q = 10)$.

$\hat{S}$ is initialized to be S and is serially updated during the Chien search stage. Starting from the 848^{th} clock cycle, the 10-bit FIFO output x_i (possibly distorted codeword) and the decoded 10-bit error vector e_i will be buffered in two 10-bit registers. At each odd-numbered clock cycle, $\hat{S}$ is updated as follows.

$$\hat{S} = \hat{S} \oplus x_{i-1} \bullet x_i \oplus (x_{i-1} \oplus e_{i-1}) \bullet (x_i \oplus e_i). \quad (23)$$

At the 1675 clock cycle, msb and $\hat{S}$ are used to re-check whether the most significant two bits are successfully cor-

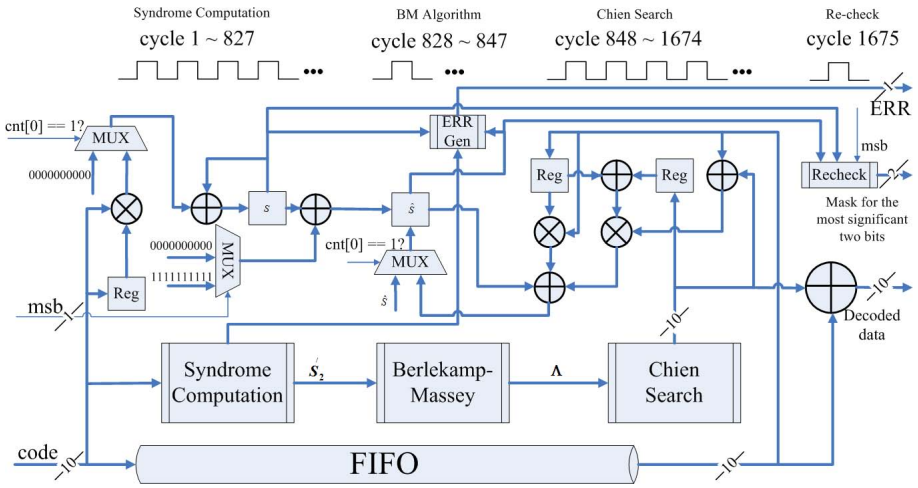


Fig. 6: Decoder architecture for the proposed $(8281, 8201, 11)$ nonlinear 5-error-correcting code

rected. A 2-bit error mask will be generated to make adjustment to these two bits according to the check results.

The decoder for the digit-error correcting code based on Theorem 3 presented in Section V and the decoder for the $(8281, 8201, 11)$ nonlinear 5-bit error correcting code are different as follows.

1) All operations of the decoder for the 5-digit error correcting code are performed in $GF(2^{10})$.
2) The 5-digit error correcting code does not require a parallel architecture. A serial design can achieve a similar decoding latency in terms of the number of clock cycles to the decoder for the $(8281, 8201, 11)$ 5-bit error correcting code with a parallelism level of 10.
3) One more block for the computation of the error magnitude is integrated into the architecture shown in Figure 6. The block is connected to the Chien search block and generates the final decoded memory contents.

The error magnitude polynomial is generated by the Berlekamp-Massey block. To reduce the hardware overhead, multipliers in $GF(2^{10})$ for the calculation of the nonlinear syndrome S are re-used to generate the error magnitude polynomial. One inverter in $GF(2^{10})$ is required to compute e_i according to Forney's algorithm (see (22)). In general, inverters in Galois field have much longer critical path than multipliers. Thus a 4-stage pipeline is added to reduce the latency of the inverter. Let $\sigma \in GF(2^{10})$, σ^{-1} can be represented as

$$\sigma^{-1} = \sigma^{2^{10}-2} = \sigma^{2^1+2^2+\cdots+2^9} \tag{24}$$

Given the fact that a 4-stage pipeline is implemented, the above function can be realized using square operations and 5 multiplications in $GF(2^{10})$. Again we re-use the multipliers in other blocks for the purpose of reducing the hardware overhead. Since the square operation is simple in $GF(2^{10})$, the inverter adds minimal area overhead and has a latency similar to the Galois filed multiplier in our design.

C. Area Overhead, Latency and Power Consumption

The area overhead, latency and the power consumption for architectures based on the six alternatives presented in Section V are shown in Table III. The designs are modelled in Verilog and synthesized in RTL Design Compiler using 45nm NANGATE library[34]. In practice the logic circuits used in NAND flash memory could be different from those used in standard digital designs. The estimation presented here is only for the purpose of investigating the increase in area, power and latency of architectures based on the proposed nonlinear multi-error correcting codes compared to architectures based on the widely used BCH codes and Reed-Solomon codes.

During the synthesis we fixed the clock rate for the encoder and the decoder and compared the area and the power consumption for architectures based on different codes. The encoders work at $1GHz$. The decoders work at a lower frequency – $400MHz$ – due to the long critical path in Berlekamp-Massey block [12]. The six alternatives require the similar latency in terms of the number of clock cycles for encoding and decoding. Due to the computation of the error magnitude and the pipeline for the inverter in the Galois field, digit-error correcting codes (Reed-Solomon, etc) need 8 more clock cycles to complete the decoding compared to bit-error correcting codes (BCH, etc).

The encoders for the digit-error correcting codes require $40\% \backsim 50\%$ more area overhead and power than the encoders for bit-error correcting codes (Figure 7 and 8) due to the fact that all operations are in $GF(2^{10})$. The decoders for digit-error correcting codes, however, require $20\% \backsim 30\%$ less overhead in area and power because of a much simpler serial architecture.

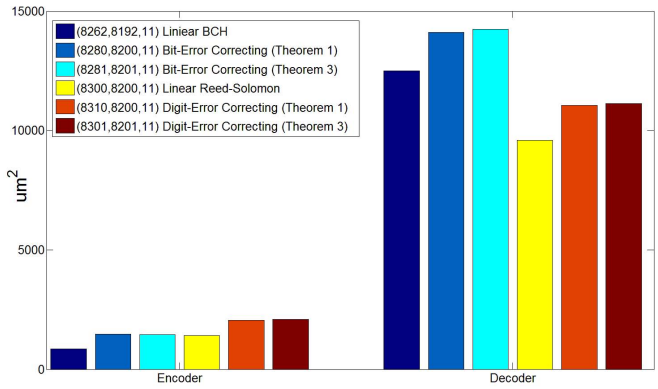
Compared to BCH codes and Reed-Solomon codes, the proposed nonlinear multi-error correcting codes need about $10\% \backsim 20\%$ more area and power in total for the encoder and the decoder and have the similar latency in terms of the number of clock cycles required to complete the encoding and decoding. The $(8281, 8201, 11)$ nonlinear 5-bit error correcting codes based on Theorem 3 (column 6 and 7 in Table III), for example, requires 17.5% more area and consumes 10.0% more

TABLE III: Comparison of the area, the latency and the power consumption of different alternatives that can correct up to 5-bit errors for the protection of MLC NAND flash memories

	Bit-Error Correcting						Digit-Error Correcting					
	BCH		Theorem 1		Theorem 3		RS		Theorem 1		Theorem 3	
	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC	ENC	DEC
Parallelism Level	10	10	10	10	10	10	1	1	1	1	1	1
Speed(Hz)	1G	400M	1G	400M	1G	400M	1G	400M	1G	400M	1G	400M
Latency(Cycles)	820	1674	821	1675	820	1675	820	1682	821	1683	820	1683
Area(μm^2)	854.4	12501.7	1468.3	14114.0	1459.8	14245.9	1407.4	9597.8	2051.7	11059.2	2094.5	11127.3
Power(mW)	0.61	3.31	0.85	3.37	0.82	3.48	0.91	2.03	1.13	2.14	1.20	2.18

power in total for the encoder and the decoder compared to the $(8262, 8192, 11)$ BCH code.

We note that the encoder and the decoder are only a very small portion in the MLC NAND flash memory chip, where the major portion is the memory cell array. Thereby the increase in area overhead for the encoder and the decoder is not significant for the reliable memory design. The power for ECC schemes is mostly consumed by the decoder. However, when there are no errors, which is the most probable case, the only active part in the decoder is the syndrome computation block. Thereby, in practice the average increase of the power consumption for the presented nonlinear multi-error correcting codes will be smaller than the data shown in Table III. Given the fact that the reliability of MLC NAND flash memories protected by the proposed nonlinear multi-error correcting codes are much higher than those protected by linear codes (Table II), the small increase in area and power is reasonable and acceptable.

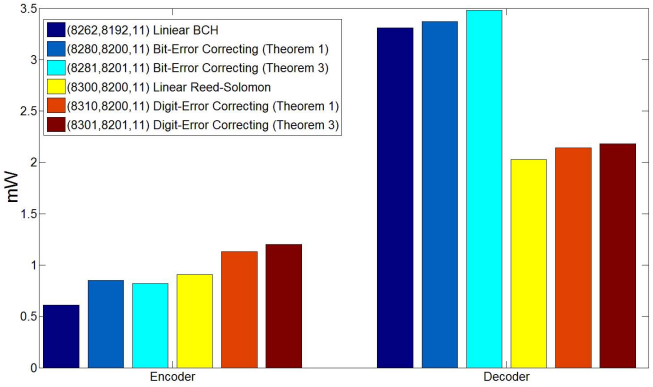


* For digit error correcting codes, n and k are regarding to bits.

Fig. 7: Comparison of the area overhead of the encoder and the decoder for different alternatives

VII. Conclusions

In this paper, the constructions of two nonlinear multi-error correcting codes are proposed. Their error correcting algorithms are presented. The proposed codes have much less undetectable and miscorrected errors than the conventional linear BCH codes and linear Reed-Solomon codes. The code constructed based on Theorem 3 can also correct some errors with multiplicity larger than its error correcting capability t without any extra overhead in area, timing and



* For digit error correcting codes, n and k are regarding to bits.

Fig. 8: Comparison of the power consumption of the encoder and the decoder for different alternatives

power consumption compared to schemes that correct only up to t errors. The beyond-t error correcting capability of the presented nonlinear multi-error correcting codes results in a further improvement of the reliability of the system.

The designs of reliable MLC NAND flash memories based on the proposed nonlinear multi-error correcting codes are presented. We compare the area, the latency and the power consumption of the proposed reliable MLC NAND flash architectures to architectures based on linear BCH codes and linear Reed-Solomon codes. The encoder and the decoder for all the alternatives are modeled in Verilog and synthesized in RTL Design Compiler. The results show that architectures based on nonlinear multi-error correcting codes can have close to zero undetectable and miscorrected errors at the cost of less than 20% increase in area overhead and power consumption compared to architectures based on the linear BCH codes and the linear Reed-Solomon codes (Table III).

References

[1] G. Atwood, A. Fazio, D. Mills, and B. Reaves, "Intel StrataFlash^{TM} memory technology overview," *Intel Technology Journal*, vol. 1, 1997.

[2] J. Cooke, "The inconvenient truths about NAND flash memory." Micron MEMCON'07 presentation, 2007.

[3] R. Dan and R. Singer, "White paper: Implementing MLC NAND flash for cost-effective, high capacity memory." M-Systems, 2003.

[4] R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti,

"Introduction to flash memory," *Proceedings of the IEEE*, vol. 91, pp. 489–502, April 2003.

[5] G. Cellere, S. Gerardin, M. Bagatin, A. Paccagnella, A. Visconti, M. Bonanomi, S. Beltrami, R. Harboe-Sorensen, A. Virtanen, and P. Roche, "Can atmospheric neutrons induce soft errors in NAND floating gate memories?," *Electron Device Letters, IEEE*, vol. 30, pp. 178–180, Feb. 2009.

[6] M. Bagatin, G. Cellere, S. Gerardin, A. Paccagnella, A. Visconti, S. Beltrami, and M. Maccarrone, "Single event effects in 1Gbit 90nm NAND flash memories under operating conditions," in *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, pp. 146–151, July 2007.

[7] F. Irom and D. Nguyen, "Single event effect characterization of high density commercial NAND and NOR nonvolatile flash memories," *Nuclear Science, IEEE Transactions on*, vol. 54, pp. 2547–2553, Dec. 2007.

[8] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *Signal Processing Systems Design and Implementation, 2006. SIPS '06. IEEE Workshop on*, pp. 303–308, Oct. 2006.

[9] R. Micheloni, R. Ravasio, A. Marelli, E. Alice, V. Altieri, A. Bovino, L. Crippa, E. Di Martino, L. D'Onofrio, A. Gambardella, E. Grillea, G. Guerra, D. Kim, C. Missiroli, I. Motta, A. Prisco, G. Ragone, M. Romano, M. Sangalli, P. Sauro, M. Scotti, and S. Won, "A 4Gb 2b/cell NAND flash memory with embedded 5b BCH ECC for 36mb/s system read throughput," in *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pp. 497–506, Feb. 2006.

[10] F. Sun, S. Devarajan, K. Rose, and T. Zhang, "Design of on-chip error correction systems for multilevel NOR and NAND flash memories," *IET Circuits, Devices and Systems*, vol. 1, no. 3, pp. 241–249, 2007.

[11] T.-H. Chen, Y.-Y. Hsiao, Y.-T. Hsing, and C.-W. Wu, "An adaptive-rate error correction scheme for NAND flash memory," in *VLSI Test Symposium, 2009. VTS '09. 27th IEEE*, pp. 53–58, May 2009.

[12] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level NAND flash memory using Reed-Solomon codes," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, pp. 94–99, Oct. 2008.

[13] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for multi-level flash memories: Correcting asymmetric limited-magnitude errors," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pp. 1176–1180, June 2007.

[14] M. Karpovsky and A. Taubin, "New class of nonlinear systematic error detecting codes," *Information Theory, IEEE Transactions on*, vol. 50, pp. 1818–1819, Aug. 2004.

[15] Z. Wang, M. Karpovsky, and K. Kulikowski, "Replacing linear Hamming codes by robust nonlinear codes results in a reliability improvement of memories," in *Dependable Systems and Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, pp. 514–523, 29 2009-July 2 2009.

[16] K. Kulikowski, Z. Wang, and M. Karpovsky, "Comparative analysis of robust fault attack resistant architectures for public and private cryptosystems," in *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC '08. 5th Workshop on*, pp. 41–50, Aug. 2008.

[17] J. L. Vasil'ev, "On nongroup close-packed codes," in *Probl.Kibernet.*, vol. 8, pp. 375–378, 1962.

[18] Z. Wang, M. G. Karpovsky, and N. Joshi, "Reliable MLC NAND flash memories based on nonlinear t-error-correcting codes," in *2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, pp. 41–50, IEEE Computer Society, 2010.

[19] Micron, *Wear-leveling techniques in NAND flash devices*, 2008.

[20] E. Yaakobi, J. Ma, A. Caulfield, L. Grupp, S. Swanson, P. H. Siegel, and J. K. Wolf, "Error correcting coding for flash memories," tech. rep., Center for Magnetic Recording Research (CMRR), 2009.

[21] M. G. Karpovsky, K. Kulikowski, and Z. Wang, "Robust error detection in communication and computation channels," in *Proceedings of Int. Workshop on Spectral Techniques*, 2007.

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

[23] M. Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *J. Complex.*, vol. 13, no. 1, pp. 180–193, 1997.

[24] Z. Wang, M. Karpovsky, and K. Kulikowski, "Design of memories with concurrent error detection and correction by nonlinear SEC-DED codes," *Journal of Electronic Testing*, vol. 26, pp. 559–580, 2010. 10.1007/s10836-010-5168-5.

[25] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. North-Holland Pub., 1977.

[26] T.-B. Pei and C. Zukowski, "High-speed parallel CRC circuits in VLSI," *Communications, IEEE Transactions on*, vol. 40, pp. 653–657, Apr 1992.

[27] J. Cho and W. Sung, "Strength-reduced parallel Chien search architecture for strong BCH codes," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 55, pp. 427–431, May 2008.

[28] H. Burton, "Inversionless decoding of binary BCH codes," *Information Theory, IEEE Transactions on*, vol. 17, pp. 464–466, Jul 1971.

[29] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 5, pp. 641–655, 2001.

[30] K. Seth, K. Viswajith, S. Srinivasan, and V. Kamakoti, "Ultra folded high-speed architectures for Reed Solomon decoders," in *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on*, pp. 4 pp.–, Jan. 2006.

[31] S. Rizwan, "Retimed decomposed serial Berlekamp-Massey (BM) architecture for high-speed Reed-Solomon decoding," in *VLSI Design, 2008. VLSID 2008. 21st*

International Conference on, pp. 53–58, Jan. 2008.

[32] Y. Chen and K. Parhi, "Small area parallel Chien search architectures for long BCH codes," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, pp. 545–549, May 2004.

[33] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE transactions on VLSI Systems*, vol. 9, pp. 641–655, 2001.

[34] "Nangate 45nm open cell library." `http://www.nangate.com`.