

# Error Correction for Partially Stuck Memory Cells

Haider Al Kim<sup>1,2</sup>, Sven Puchinger<sup>1</sup>, Antonia Wachter-Zeh<sup>1</sup>

<sup>1</sup>Institute for Communications Engineering, Technical University of Munich (TUM), Germany

<sup>2</sup>Electrical and Communication Engineering, University of Kufa, Iraq

Email: {haider.alkim, sven.puchinger, antonia.wachter-zeh}@tum.de

**Abstract**—We present code constructions for masking  $u$  partially stuck memory cells with  $q$  levels and correcting additional random errors. The results are achieved by combining the methods for masking and error correction for stuck cells in [1] with the masking-only results for partially stuck cells in [2]. We present two constructions for masking  $u < q$  cells and error correction: one is general and based on a generator matrix of a specific form. The second construction uses cyclic codes and allows to efficiently bound the error-correction capability using the BCH bound. Furthermore, we extend the results to masking  $u \geq q$  cells. For  $u > 1$  and  $q > 2$ , all new constructions require less redundancy for masking partially stuck cells than previous work on stuck cells, which in turn can result in higher code rates at the same masking and error correction capability.

**Index Terms**—flash memories, phase change memories, (partially) stuck cells, error correction, defective cells, partitioned cyclic codes, BCH code.

## I. INTRODUCTION

The dominance of non-volatile memories such as PCMs (phase change memories) as memory solutions for a variety of applications has become significant due to their advantages as permanent storage devices. The advantages of these memories are their rapid increase in capacity plus their ability as multi-level technologies. For these reasons, their cost has been reduced strongly in the last years. However, reliability issues make it necessary to suggest new sophisticated coding and signal processing solutions. PCM cells can hold two states: an amorphous state and several crystalline states. Non-defective memory cells can switch between their main states (amorphous and crystalline). However, due to the cooling and heating processes of the cells, PCMs may face failures in changing their states, and therefore the cells can hold only one phase and they become *stuck* [3]–[6].

This means that the cell’s charge is trapped in the cell, and it cannot change its status to be re-written. To deal with the stuck positions, a mechanism called *masking* is used. Masking finds a codeword that holds the same levels as in the stuck positions, and can therefore be placed properly on the memory. For multi-level PCMs, since the crystalline state can be programmed into partial states, the cell may be stuck in this level or in one of its sub-levels (level higher than 0). If the cell can only represent levels greater or equal to a reference level  $s > 0$ , it is called a *partially stuck cell* [2]. For multi-level PCMs, the case  $s = 1$  is particularly important since this means that a cell cannot reach the amorphous state anymore, but all partially crystalline ones, cf. [2]. Similarly, (partially) stuck cells can occur in flash

memories. Flash memory stores information by charging it electronically. If charge is trapped inside one cell at a certain level (one of the main levels or any intermediate levels), the ways to rewrite on again are by increasing the trapped level or by erasing one whole block. However, erasing whole block reduces the lifetime of these memory devices. Figure 1 shows the general idea of the (partially) stuck memory cells. On the other hand, it happens that due to manufacturing defects, cells can only hold lower levels causing the reverse problem as partially stuck cells.

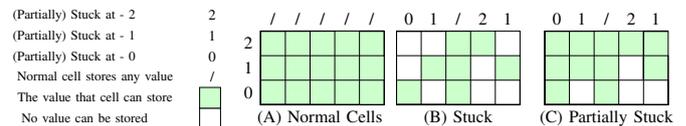


Figure 1. General idea of (partially) stuck memory cells. In this figure, there are  $n = 5$  cells with  $q = 3$  possible levels. The stuck levels are (0, 1 or 2). In case (A), normal cells can store any of the three values. In the stuck scenario as shown in case (B), the stuck cell can store only the exact stuck level  $s$ . It is more flexible in case (C) (partially stuck scenario). A cell that is partially stuck at 0 can store any value as a non-defective cell. Partially stuck cells at level  $s \geq 1$  can store one level or more. This paper only deals with partially stuck at 1 cells.

### A. Related Work

In [1], code constructions for masking *stuck memory cells* were proposed. In addition to masking the defect cells, it is possible to correct errors that occur during the storing and reading processes. A generator matrix of a specific form was constructed for this purpose. Moreover, in [1], a partitioned cyclic code and partitioned BCH were proposed to mask stuck cells and correct errors. However, the required redundancy for masking more than one cell is greater than one symbol. Later, an asymptotic optimal analysis for  $n \rightarrow \infty$  was proposed to mask defects of (fixed<sup>1</sup> [7], linearly increasing<sup>2</sup> [8]) multiplicity that need a redundancy of at least the number of defects. Besides masking the defects, the construction in [8, Section 5] can correct errors with overall redundancy  $r(n, \rho, \tau) = \rho + o(n)$ , where  $\rho$  denotes the number of defects and  $\tau$  denotes the number of errors.  $o(n)$  is the required redundancy to correct errors.

In [2], improvements on the redundancy necessary for masking *partially stuck memory cells* are achieved, and lower and upper bounds are derived by the constructions. However, the paper does not consider error correction in addition to masking.

<sup>1</sup>An asymptotically optimal class of codes was proposed to correct  $\rho$  defects ( $\rho = \text{const value}$ ) regardless of  $n \rightarrow \infty$ , where  $n$  is the code length.

<sup>2</sup>Asymptotically optimal linear codes were proposed to correct  $\rho$  defects ( $\rho = \alpha n$ ), where  $\alpha = \text{constant value}$  and  $\rho = \text{number of defects}$  that is linearly increasing while  $n \rightarrow \infty$ , i.e. if  $n$  is doubled then  $\rho$  is doubled.

This work has received funding from the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under Grant No. WA3907/1-1 and from the German Israeli Project Cooperation (DIP) grant no. KR3517/9-1. Al Kim has received funding from the German Academic Exchange Service (Deutscher Akademischer Austauschdienst, DAAD) under the support program ID 57381412.

## B. Our Contribution

In this paper, we combine the methods of [1] and [2]. We obtain code constructions for combined error correction and masking partially stuck cells. Compared to the stuck-cell case in [1], we reduce the redundancy necessary for masking, similar to the results in [2].

In contrast to [2], however, we are able to correct additional random errors. Similar to the main part of [2], this paper deals with partially-stuck-at-1 cells, i.e.,  $s = 1$  (recall that this is the typical for PCMs, cf. [2]), but the results are extendable to arbitrary  $s$  similar to [2, Section VII].

On the other hand, compared to the stuck cell model in [8, Section 5], it is not clear if  $r(n, \rho, \tau) = \rho + o(n)$  is higher or lower than our overall redundancy shown in Theorem 4. This is because we do not consider an asymptotic analysis in this paper for the *partially stuck* cells with errors correction model. However, under the assumption that both of them require  $o(n)$  redundancy to correct errors, our overall redundancy shown in Theorem 4 is lower. The reason is that [8, Section 5] uses  $\rho$  check symbols to mask the *stuck* cells while our construction uses less than  $\rho$  as a redundancy necessary to mask *partially stuck* cells. Moreover, our constructions can correct a certain number of errors and mask a certain number of *partially stuck* cells, while [8, Section 5] proposed  $o(n)$  redundancy for error correction that is negligible, i.e.  $o(n) \rightarrow 0 \iff n \rightarrow \infty$ .

## II. PRELIMINARIES

### A. Notations

For a prime power  $q$ , let  $\mathbb{F}_q$  denote the finite field of order  $q$  and  $\mathbb{F}_q[x]$  be the set of all univariate polynomial with coefficients in  $\mathbb{F}_q$ . Write  $[f] = \{0, 1, \dots, f-1\}$ . Let  $k_1$  be the number of information symbols,  $l$  be the required symbol(s) for masking,  $r$  be the required redundancy for error correction,  $t$  be the number of errors,  $u$  be the number of (partially) stuck cells. Let  $s_{\phi_i}$  denote the (partially) stuck level at any position, where  $i \in [u]$ . Let  $n$  be the code length and also the memory size.

### B. Definitions

1) *Stuck and Partially Stuck Cells*: A cell is called *stuck at level  $s$* , where  $s \in [q]$ , if it cannot change its value and always stores the value  $s$ . A cell is called *partially stuck at level  $s$* , where  $s \in [q]$ , if it can only store values which are at least  $s$ . If a cell is partially stuck at 0, it is a non-defect cell which can store any of the  $q$  levels [2].

2) *(u, t)-PSMC*: An  $(n, M)_q$   $(u, t)$ -*partially-stuck-at-masking code*  $\mathcal{C}$  is a coding scheme consisting of an encoder  $\mathcal{E}$  and decoder  $\mathcal{D}$ . The input of the encoder  $\mathcal{E}$  is

- the set of locations of  $u$  partially stuck cells  $\phi = \{\phi_0, \phi_1, \phi_2, \dots, \phi_{u-1}\} \subseteq [n]$ ,
- the partially stuck levels  $s_{\phi_0}, s_{\phi_2}, \dots, s_{\phi_{u-1}} \in [q]$ , and
- a message  $\mathbf{m} \in \mathcal{M}$ , where  $\mathcal{M}$  is a message space of cardinality  $|\mathcal{M}| = M$

It outputs a vector  $\mathbf{c} \in \mathbb{F}_q^n$  which fulfills  $c_{\phi_i} \geq s_{\phi_i}$  for all  $i = 1, \dots, u$ . The decoder is a mapping that takes  $\mathbf{c} + \mathbf{e} \in [q]^n$  as input and returns the correct message  $\mathbf{m}$  for all error vectors  $\mathbf{e}$  of Hamming weight at most  $t$ .

3) *Q-ary Cyclic Code*: a  $q$ -ary cyclic code of length  $n$ , dimension  $k$ , and minimum distance  $d$  is denoted as an  $[n, k, d]_q$  code  $\mathcal{C}$ . It has a generator polynomial  $g(x)$  of degree  $n - k$  with roots in  $\mathbb{F}_{q^m}$ , where  $n$  divides  $q^m - 1$ .

A cyclotomic coset  $M_a$  is given by:

$$M_a := \{a \cdot q^j \pmod n, \forall j = 0, 1, \dots, n_a - 1\}, \quad (1)$$

where  $n_a$  is the smallest integer such that  $a \cdot q^{n_a} \equiv a \pmod n$ . Let  $\alpha \in \mathbb{F}_{q^m}$  be a primitive  $n^{\text{th}}$  root of unity in  $\mathbb{F}_{q^m}$ . The minimal polynomial of an element  $\alpha^a$  is given by:

$$M^{(a)}(x) := \prod_{b \in M_a} (x - \alpha^b) \quad (2)$$

Although the factors  $(x - \alpha^b)$  are in  $\mathbb{F}_{q^m}[x]$ , minimal polynomials have coefficients in the small field  $\mathbb{F}_q$ , i.e.,  $M^{(a)}(x) \in \mathbb{F}_q[x]$ . The defining set  $D_c$  of a  $q$ -ary cyclic code  $\mathcal{C}$  with parameters  $[n, k, d_1]_q$  is the set containing the indices  $b$  of the zeros  $\alpha^b$  of the generator polynomial  $g(x)$ . If  $a \in D_c$ , then we have  $M_s \subseteq D_c$ , and hence,  $D_c$  is a union of cyclotomic cosets  $M_{a_1}, \dots, M_{a_w}$  for some  $w$ , i.e.

$$D_c := \{b : g(\alpha^b) = 0\} = M_{a_1} \cup M_{a_2} \cup M_{a_3} \dots \cup M_{a_w}. \quad (3)$$

The generator polynomial  $g(x) \in \mathbb{F}_q[x]$  of the code  $[n, k, d_1]$  of degree  $r = n - k$  is thus given by

$$g(x) = \prod_{a \in D_c} (x - \alpha^a) = \prod_{b=1}^w M^{(a_b)}(x). \quad (4)$$

For any cyclic code, there is a parity-check polynomial:

$$h(x) = \frac{(x^n - 1)}{g(x)} = \prod_{a \in [n] \setminus D_c} (x - \alpha^a). \quad (5)$$

The minimum distance of a cyclic code is at least its BCH bound, which is the number of consecutive elements in  $D_c$  plus one.

## III. CODES FOR (PARTIALLY) STUCK CELLS

### A. Masking Partially Stuck and Correcting Additional Random Errors using a Generator Matrix Construction

The goal of this paper is to find a code construction that can store information in a memory with some partially stuck cells and additionally can correct errors. For the masking process and according to [2], we need only a single redundancy symbol if  $u < q$  and  $(s_{\phi_i} = 1)$ . However, the work in [2] does not consider correcting additional random errors. The following theorem introduces a code construction using a generator matrix with a specific form that masks partially stuck cells and corrects errors.

**Theorem 1.** Let  $u \leq \min\{n, q - 1\}$ . Assume there is an  $[n, k, d \geq 2t + 1]_q$  code  $\mathcal{C}$  with a  $k \times n$  generator matrix of the following form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{k_1 \times 1} & \mathbf{I}_{k_1} & \mathbf{P}_{k_1 \times r} \\ 1 & \mathbf{1}_{1 \times k_1} & \mathbf{1}_{1 \times r} \end{bmatrix},$$

where:

- $\mathbf{G}_1$  is a  $k_1 \times n$  generator matrix of an  $[n, k_1]_q$  code  $\mathcal{C}_1$ .
- $k_1 = n - 1 - r$ .
- $k = k_1 + 1$ .
- $\mathbf{P} \in \mathbb{F}_q^{k_1 \times r}$ .

By using Algorithm 1 and 2, this code is a  $(u, t)$ -PSMC which can mask  $u$  partially stuck memory cells, and can correct  $t$  additional random errors while storing  $k_1 = n - r - 1$  information symbols.

---

**Algorithm 1: Encoding**

---

**Input:**

- Message:  $\mathbf{m} = (m_0, m_1, \dots, m_{k_1-1}) \in \mathbb{F}_q^{k_1}$
- Positions of partially stuck cells:  $\phi$

- 1 Compute the message vector  $\mathbf{w} = \mathbf{m} \cdot \mathbf{G}_1$
- 2 Find  $v \in \mathbb{F}_q$  such that  $w_{\phi_i} \neq v$
- 3  $z_0 \leftarrow q - v \equiv -v \pmod{q}$
- 4 Compute the masking vector  $\mathbf{d} = z_0 \cdot \mathbf{G}_0$
- 5 Compute  $\mathbf{c} = (\mathbf{w} + \mathbf{d}) \pmod{q}$

**Output:** Codeword  $\mathbf{c} \in \mathbb{F}_q^n$ 

---

---

**Algorithm 2: Decoding**

---

**Input:**

- Retrieve  $\mathbf{y} = \mathbf{c} + \mathbf{e}$ ,  $\mathbf{y} \in \mathbb{F}_q^n$

- 1  $\hat{\mathbf{c}} \leftarrow$  decode  $\mathbf{y}$  in  $\mathcal{C}$
- 2  $\hat{z}_0 \leftarrow$  first entry of  $\hat{\mathbf{c}}$
- 3  $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1, \dots, \hat{w}_{n-1}) \leftarrow (\hat{\mathbf{c}} - \hat{z}_0 \cdot \mathbf{G}_0) \pmod{q}$
- 4  $\hat{\mathbf{m}} \leftarrow (\hat{w}_1, \dots, \hat{w}_{k_1})$

**Output:** Message vector  $\hat{\mathbf{m}} \in \mathbb{F}_q^{k_1}$ 

---

*Proof.* Since in partially stuck cells the stuck level  $s_{\phi_i} \geq 1$  has to be masked, the output codeword  $\mathbf{c}$  must match the partially stuck positions:

$$c_{\phi_0}, c_{\phi_1}, \dots, c_{\phi_{u-1}} \geq 1. \quad (6)$$

Since  $u < q$ , there is at least one value  $v \in \mathbb{F}_q$  such that  $w_{\phi_0}, w_{\phi_1}, \dots, w_{\phi_{u-1}} \neq v$ . Thus, we choose  $z_0 = q - v \equiv -v \pmod{q}$ . Therefore  $c_{\phi_i} = w_{\phi_i} + z_0 \equiv (w_{\phi_i} - v) \pmod{q} \neq 0$  and (6) is satisfied.

The decoder (Algorithm 2) gets  $\mathbf{y}$ , which is  $\mathbf{c}$  corrupted by at most  $t$  errors. Since  $\mathcal{C}$  has minimum distance  $d \geq 2t + 1$ , we can correct these errors and obtain  $\hat{\mathbf{c}}$ . Due to the structure of the matrix  $\mathbf{G}$ , the first position of  $\hat{\mathbf{c}}$  equals the masking value  $z_0$ . Hence, we can compute  $\hat{\mathbf{w}} = \mathbf{w}$  (cf. Algorithm 2) and the message vector  $\hat{\mathbf{m}} = \mathbf{m}$ .  $\square$

Theorem 1 gives an extension of [1, Theorem 1] and [2, Theorem 4]. It combines [1] and [2] to provide a code construction that can mask partially stuck cells and correct errors. The required redundancy is a single symbol for masking plus the redundancy for the code  $\mathcal{C}_1$ . In comparison, [1, Theorem 1] requires at least

$$\min\{n - k : \exists [n, k, d]_q \text{ code with } d > u\} \geq u.$$

redundancy symbols to mask  $u$  cells, where the inequality follows directly from the Singleton bound.

The code construction of Theorem 1 is based on the matrix  $\mathbf{P}$ , which is hard to find in general. One way to construct such a matrix is to start with a generator matrix of a well-known code that contains the all-one vector (e.g. certain cyclic codes, see the next sections) and has a large enough minimum distance. It is easy to see that the generator matrix can be transformed into the form of Theorem 1 by elementary row operations and column permutations.

### B. Masking Partially Stuck and Correcting Additional Random Errors using a Partitioned Cyclic Code Construction

This section generalizes the construction of [1, Theorem 2]. Our construction uses a so-called *partitioned cyclic code* as in [1], but requires only a single redundancy symbol  $l = 1$  for

the masking operation similar to Theorem 4 and Algorithm 3 in [2]. Additionally, it corrects errors by selecting a generator polynomial  $g_1(x)$  of degree  $r$ , where  $0 \leq r < n - 1$ . As the construction in the previous section, this new construction results in a reduced redundancy  $l$  compared to masking stuck cells in [1, Theorem 2]. Compared to Theorem 1 in the previous section, the cyclic construction directly implies a constructive strategy how to choose a code of a certain minimum distance.

**Construction 1.** Let  $u \leq \min\{n, q - 1\}$ . Let  $\mathcal{C}_0$  be an  $[n, n - 1, \delta_0]_q$  code with parity-check polynomial  $g_0(x) = 1 + x + x^2 + \dots + x^{n-1}$ . Let:

$$c_0(x) = z_0 \cdot g_0(x).$$

Let  $\mathcal{C}_1$  be an  $[n, n - r, \delta_1]_q$  code where  $g_1(x)$  is its monic generator polynomial of degree  $r$  that divides  $g_0(x)$ , and  $h_1(x)$  is its parity-check polynomial. Then let  $\mathcal{C}$  be an  $[n, n - r - 1, \delta_1, \delta_0]_q$  partitioned cyclic code built from  $\mathcal{C}_1$  and  $\mathcal{C}_0$  with the following encoding rule:

$$c(x) = m(x) \cdot g_1(x) + z_0 \cdot g_0(x),$$

where  $m(x) \in \mathbb{F}_q[x]$  is the message polynomial of degree  $< n - r - 1$  and  $z_0$  is a scalar.

**Theorem 2.** If  $u \leq \min\{n, q - 1\}$ , Construction 1 provides an  $(n, M = q^{n-r-1})_q (u, t)$ -PSMC with redundancy of  $1 + r$  symbols by using Algorithms 3 and 4.

---

**Algorithm 3: Encoding**

---

**Input:**

- Message:  $m(x) \in \mathbb{F}_q[x]$  of degree  $< n - r - 1$
- Positions of partially stuck cells:  $\phi$

- 1  $n = q^m - 1$ ,  $m$  is an integer
- 2 Let  $g_0(x) = 1 + x + x^2 + \dots + x^{n-1}$  and  $h_0(x) = x - 1$
- 3 Select  $g_1(x) \mid g_0(x)$  of degree  $r$ , where  $0 \leq r < n - 1$
- 4  $c_1(x) = c_{10} + \dots + c_{1n-2} \cdot x^{n-2} \leftarrow m(x) \cdot g_1(x)$
- 5 Assign  $v \in \mathbb{F}_q$  such that  $c_{1\phi_0}, c_{1\phi_1}, \dots, c_{1\phi_{u-1}} \neq v$
- 6 Find  $z_0 = -v \pmod{q}$
- 7  $c_0(x) = z_0 \cdot g_0(x)$
- 8  $c(x) = c_1(x) + c_0(x) \pmod{(x^n - 1)}$

**Output:** Codeword  $c(x) \in \mathbb{F}_q[x]$  of degree  $\leq n - 1$ 

---

---

**Algorithm 4: Decoding**

---

**Input:** Retrieve  $y(x) = c(x) + e(x)$ 

- 1  $\hat{c}(x) \leftarrow$  Decode  $y(x)$  in the code generated by  $g_1(x)$ .
- 2  $\hat{m}(x) \leftarrow \hat{c}(x) \pmod{g_0(x)}$

**Output:** Message  $\hat{m}(x) \in \mathbb{F}_q[x]$  of degree  $< n - r - 1$ 

---

*Proof.* Algorithm 3 shows the encoding process for the partitioned cyclic code construction. Let  $m(x) = m_0 + m_1 \cdot x + m_2 \cdot x^2 + \dots + m_{n-r-2} \cdot x^{n-r-2}$ . Algorithm 3 calculates  $c_1(x)$  in Step 4 of degree  $< n - 1$ . Since  $u < q$ , there is at least a single value  $v \in \mathbb{F}_q$  such that the coefficients of  $c_1(x)$  in the partially stuck positions  $c_{1\phi_0}, c_{1\phi_1}, \dots, c_{1\phi_{u-1}} \neq v$ . So we choose  $z_0 = -v \pmod{q}$  as shown in Step 5. Note that the value of  $z_0$  always appears in  $c(x)$ . That is, the last memory location stores  $z_0$ . Since in partially stuck cells the stuck level  $s_{\phi_i} \geq 1$ , the coefficients of  $c(x)$  must match the partially stuck positions as in (6). Equation (6) is also satisfied because  $c_{\phi_i} = c_{1\phi_i} + z_0 \equiv (c_{1\phi_i} - v) \pmod{q} \neq 0$ .

Algorithm 4 decodes the retrieved polynomial  $y(x)$ . First, decode  $y(x)$  with the code generated by  $g_1(x)$ . We can correct

a random error  $e(x)$  (if any) by taking the module operation  $\text{mod } g_1(x)$ . We must choose  $\hat{e}(x) \in \mathbb{F}_q^n$  which minimizes  $e(x)$  and satisfies  $\hat{e}(x) \text{ mod } g_1(x) = e(x) \text{ mod } g_1(x)$ .

Then, the algorithm performs the unmasking process to find  $\hat{m}(x)$ . If  $\hat{m}(x) = m(x)$  of degree  $< n - r - 1$ , the decoding is successful. Taking  $\text{mod } g_0(x)$  for  $\hat{c}(x)$  gives  $\hat{m}(x) = m(x)$ :

$$\begin{aligned} \hat{c}(x) &= \hat{m}(x) \cdot g_1(x) + z_0 \cdot g_0(x) \rightarrow \\ \hat{m}(x) &= \frac{\hat{c}_1(x) \text{ mod } g_0(x)}{g_1(x)} \\ \rightarrow \hat{m}(x) &= m(x) \end{aligned}$$

□

**Remark 1.** We briefly comment on how to choose  $r$  in Theorem 2. To store  $n - 1$  information symbols, choose  $r = 0$  and  $g_1(x) = x^0 = 1$  so that  $\mathcal{C}$  becomes  $\mathcal{C}_0$ . For combined masking and error correction, choose  $r$  between  $1 \leq r < n - 1$ . However, the chosen value of  $r$  should be the smallest value such that an  $[n, n - r, \delta_1]_q$  code exists.

### C. Further Decreasing the Redundancy in Construction 1

According to [2, Construction 3], it is possible to further decrease the required redundancy necessary for masking to be  $< 1$  in Construction 1 if the  $v$  value is chosen from a small subset of  $[q]$  such that  $v \in [u + 1]$  and  $v \notin w_{\phi_i} \text{ mod } (u + 1)$ . Since the set  $\{w_{\phi_0} \text{ mod } (u + 1), w_{\phi_1} \text{ mod } (u + 1), w_{\phi_2} \text{ mod } (u + 1), \dots, w_{\phi_{u-1}} \text{ mod } (u + 1)\} \text{ mod } (u + 1)$  has the cardinality  $u$  and there are  $u + 1$  possible values to choose from, we can always find  $v \in [u + 1]$ . Thus, the stored information  $k_1$  increases by  $\log_q \lfloor \frac{q}{u+1} \rfloor$  which is the amount that the required redundancy for masking decreases  $1 - \log_q \lfloor \frac{q}{u+1} \rfloor$ . Construction 1 will be modified such that  $v$  and the coefficients of  $g_0(x)$ ,  $h_0(x)$ , and  $g_1(x) \in [u + 1]$  (from a small subset of  $q$ ), while  $z_0 \in [q]$  and  $m \in \mathbb{F}_q^{k_1}$ . We show the improvement in Construction 1 for  $k_1$  and  $l$  by  $k_1^*$  and  $l^*$  columns in Table I. We choose  $q = 6$  and  $u + 1 = 3$  to show the improvement for the same  $n$  length.

### D. Bounds for Partitioned Cyclic Code

As aforementioned, we use a partitioned cyclic  $\mathcal{C}$  (involves  $\mathcal{C}_1$  and  $\mathcal{C}_0$ ) code to mask partially stuck memory cells and correct additional random errors. However, to derive bounds for the minimum distance for the error correction part of the

code  $\mathcal{C}_1$ , we apply the BCH bound. In the *partitioned* cyclic code, there is a pair of designed distances as  $(\delta_1, \delta_0)$ , where  $\delta_1$  is the designed distance of  $\mathcal{C}_1$  with parameters  $[n, k_1 + 1, \delta_1]_q$  and  $\delta_0$  is the designed distance for the masking part of the code  $\mathcal{C}_0$  with parameters  $[n, k_1 + r, \delta_0]_q$ .

The lower bounds are the pair of designed distances  $(\delta_1, \delta_0)$  and they are computed as shown in the following:

- $\delta_0 = 2$ , use a  $[n, n - 1, 2]_2$  Single Parity Check Code.
- $\delta_1$  follows always the chosen degree  $r$  of  $g_1(x)$  based on the given lower bound in [1, Appendix]:

$$r \leq m \left\lceil \frac{\delta_1 - 1}{2} \right\rceil. \quad (7)$$

In the following section, we present Table I of length  $n = 8$  to compare ternary code for partially stuck memory cells to stuck cells with error correction as in [1].

### E. Table of Ternary Code with Code Length ( $n = 8$ )

Table I presents the maximum amount of information to be stored and the maximum errors that can be corrected. Comparing Table I to Table II from [1], an improvement is shown in Table I since only one parity symbol for masking is used. For a ternary code  $n = q^m - 1 = 8$ , for  $m = 2$  and  $q = 3$ ,  $(x^8 - 1)$  factors are:

$$(x + 1) \cdot (x^2 + 2x + 2) \cdot (x^2 + 1) \cdot (x + 2) \cdot (x^2 + x + 2) \equiv M^{(0)}(x) \cdot M^{(1)}(x) \cdot M^{(2)}(x) \cdot M^{(4)}(x) \cdot M^{(5)}(x) \text{ respectively.}$$

Although the overall redundancy  $r + l = 6$  used as shown in item 6 in Table I and item 5 in Table II, longer  $r$  in Table I increases the probability to correct more errors. This is an improvement from [1]. Furthermore, Table I shows more flexibility in storing  $k_1$  symbols as shown in item 2 where  $k_1 = 5$ . For longer code length  $n$ , our construction is more likely to have longer consecutive cyclotomic cosets and accordingly higher minimum distance, thus, can correct more errors than [1].

## IV. NEW CODES FOR PARTIALLY STUCK-AT CELLS $s_{\phi_i} = 1, q \leq u < n$

The masking technique in the previous section only guarantees successful masking up to a number of  $u < q$  partially stuck-at-1 cells. In the following, we study the problem of masking at least  $q$  cells. First, we determine the probability that masking is still possible with Construction 1 for random

Table I  
TERNARY CODES FOR PARTIALLY STUCK-AT-1 MEMORY CELL

#	$k_1$	$k_1^*$	$l$	$l^*$	$r$	$\delta_0$	$\delta_1$	$t$	$h_0(x)$	$g_1(x)$	Comment
1	6	6.387	1	0.613	1	2	2	0	$M^{(4)}(x)$	$M^{(0)}(x)$	This is masking only with one redundancy symbol.
2	5	5.387	1	0.613	2	2	2	0	$M^{(4)}(x)$	$M^{(5)}(x)$	Flexibility in storing a desired $k$ information symbols.
3	4	4.387	1	0.613	3	2	3	1	$M^{(4)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x)$	Same as Table II in assigning factors $\rightarrow$ same parameters.
4	3	3.387	1	0.613	4	2	3	1	$M^{(4)}(x)$	$M^{(1)}(x) \cdot M^{(2)}(x)$	Same as Table II in assigning factors $\rightarrow$ same parameters.
5	2	2.387	1	0.613	5	2	5	2	$M^{(4)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x) \cdot M^{(2)}(x)$	Same as Table II in assigning factors $\rightarrow$ same parameters.
6	2	2.387	1	0.613	5	2	3	1	$M^{(4)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x) \cdot M^{(5)}(x)$	No change in $\delta_1$ , however it is more likely to be increased.
7	1	1.387	1	0.613	6	2	4	1	$M^{(4)}(x)$	$M^{(1)}(x) \cdot M^{(2)}(x) \cdot M^{(5)}(x)$	Same as Table II, but it tends to be able to correct more errors.

Table II  
TERNARY CODES FOR STUCK-AT MEMORY [1]

#	$k_1$	$l$	$r$	$\delta_0$	$\delta_1$	$t$	$h_0(x)$	$g_1(x)$	Comment
1	6	1	1	2	2	0	$M^{(4)}(x)$	$M^{(0)}(x)$	Masking only.
2	4	1	3	2	3	1	$M^{(4)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x)$	Same as Table I in assigning factors $\rightarrow$ same parameters.
3	3	1	4	2	4	1	$M^{(4)}(x)$	$M^{(1)}(x) \cdot M^{(2)}(x)$	Same as Table I in assigning factors $\rightarrow$ same parameters.
4	2	1	5	2	5	2	$M^{(4)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x) \cdot M^{(2)}(x)$	Same as Table I in assigning factors $\rightarrow$ same parameters.
5	2	3	3	3	3	1	$M^{(4)}(x) \cdot M^{(5)}(x)$	$M^{(0)}(x) \cdot M^{(1)}(x)$	$l = 3$ less flexibility in chosen $r$ .
6	1	3	4	3	4	1	$M^{(4)}(x) \cdot M^{(5)}(x)$	$M^{(1)}(x) \cdot M^{(2)}(x)$	Same as Table I. However, less likely to correct more errors.

partially stuck positions. Based on [2, Construction 4], we propose a method to further increase the guaranteed number of masked cells at the cost of a masking redundancy of more than one symbol.

#### A. Probabilistic Masking

We determine the probability that for a random message, masking is possible for a number of stuck positions  $u \geq q$  if the code constructions in Theorem 1 and Theorem 2 are used. This *probabilistic masking* approach enables us to use a row of memory with a certain probability even if more than  $q - 1$  partially stuck cells are present.

**Theorem 3.** *Let  $\mathbf{G}$  be as in Theorem 1,  $q \leq u \leq k_1$ ,  $s_{\phi_i} = 1$ , and  $\phi_1, \dots, \phi_u$  such that the columns of  $\mathbf{G}$  indexed by the  $\phi_i$  are linearly independent. For a message  $\mathbf{m} \in \mathbb{F}_q^{k_1}$  that is drawn uniformly at random, the probability that we can mask the word is*

$$P(q, u) = \frac{\sum_{i=1}^q (-1)^{i+1} \binom{q}{i} \cdot (q-i)^u}{q^u}$$

*Proof.* The code discussed in Theorem 1 guarantees to mask  $u < q$  partially stuck cells since  $u$  values at the stuck positions of the intermediate codeword  $\mathbf{w}$  (after encoding only the message) do not cover the entire alphabet  $\{0, \dots, q-1\}$ . Hence, we can add a constant to all of them in order to prevent the codeword values to be 0 in the stuck positions. For the probabilistic case, we can apply the same arguments, but we need to derive a probability that the intermediate codeword values at the partially stuck positions do not constitute the entire alphabet.

Due to the assumption on the linear independence of  $\mathbf{G}$ 's columns, the vector  $(w_{\phi_1}, \dots, w_{\phi_u})$  is uniformly distributed on  $\mathbb{F}_q^u$ . The formula for  $P(q, u)$  uses the inclusion-exclusion principle [9] to count the relative number of vectors in  $\mathbb{F}_q^u$  that exclude at least one field element.  $\square$

The following example illustrates that the probability that masking is successful can be quite large.

*Example.* Let  $q = 3$ ,  $n = 8$ ,  $r = 0$  and  $v \in [q]$ , the probability in which we can mask  $u = n - 1$  partially stuck-at-1 memory cells because the values at the partially stuck positions are from a set of size at most  $q - 1$  is:

$$\frac{3 \cdot (3-1)^7 - \binom{3}{2} \cdot (3-2)^7 + \binom{3}{3} \cdot (3-3)^7}{3^7} = 0.17.$$

This ratio will be 0.77 if  $u = q$  and clearly it is 1 if  $u < q$ . Let  $\mathbf{m} = (2002220) \in \mathbb{F}_3^8$  so the augmented message vector  $\mathbf{w} = (02002220) \in \mathbb{F}_3^8$ . Let the partially stuck positions be  $w_{\phi_0}, w_{\phi_1}, w_{\phi_2}, \dots, w_{\phi_u}$  for all  $i \in [u]$  and  $u = n - 1$ . Thus,  $v = 1$  and  $z_0 = 2$ . The output from Algorithm 1 in Theorem 1 is:

$$\mathbf{c} = (02002220) + (22222222) = (21221112).$$

As it is shown from the output vector, the partially stuck-at-1 positions for all  $u$  are masked because they fulfill the following condition:

$$w_{\phi_i} + z_0 \pmod{q} \geq 1, \text{ where } q \leq u < n.$$

**Remark 2.** *The assumption in Theorem 3 that the columns of  $\mathbf{G}$  indexed by the partially stuck positions are linearly independent is fulfilled for most known codes with high probability if  $u \leq k_1$ , especially if  $u \ll k_1$ . For dependent columns, it becomes harder to count the number of vectors that cover not the entire alphabet since  $(w_{\phi_1}, \dots, w_{\phi_u})$  is uniformly distributed on a subspace of  $\mathbb{F}_q^{k_1}$ .*

#### B. Masking Up to $u \leq q + d_0 - 3$ Partially Stuck-At Cells

Construction 4 in [2] masks  $q \leq u \leq n$  partially stuck-at-1 cells. It is a generalization of the all-one vector as stated [2, Theorem 4] because if  $d_0 = 2$ ,  $u \leq q - 1$ . Theorem 1 in this paper provides a construction to mask and correct errors using a matrix with specific form that has  $(\mathbf{G}_1$  and  $\mathbf{H}_0)$ . Extending Theorem 1 by using the parity-check matrix in [2, Construction 4] instead of  $\mathbf{G}_0$  given in Theorem 1 will provide a solution to mask  $u \leq q + d_0 - 3$  and correct  $t$  errors. The result is formulated in Theorem 4.

**Theorem 4.** *Let  $u \leq q + d_0 - 3$  and  $s_{\phi_i} = 1$ . Assume there is an  $[n, k, d_0 \geq u - q + 3, d_1 \geq 2t + 1]_q$  code  $\mathcal{C}$  with a  $k \times n$  generator matrix of the following form:*

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{H}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{k_1 \times l} & \mathbf{I}_{k_1} & \mathbf{P}_{k_1 \times r} \\ & & \mathbf{H}_0 \end{bmatrix}$$

where:

- $\mathbf{G}_1$  is a  $k_1 \times n$  generator matrix of an  $[n, k_1]_q$  code  $\mathcal{C}_1$ .
- For simplicity  $\mathbf{H}_0$  is a systematic  $l \times n$  parity-check matrix of an  $[n, k_1 + r, d_0]_q$  code  $\mathcal{C}_0$ .
- $k_1 = n - l - r$ .
- $k = k_1 + l$ .
- $\mathbf{P} \in \mathbb{F}_q^{k_1 \times r}$ .

By using Algorithm 5 and 6, this code is a  $(u, t)$ -PSMC which can mask any  $u \leq n$  partially stuck memory cells, and can correct  $t$  additional random errors while storing  $k_1 = n - r - l$  information symbols.

---

#### Algorithm 5: Encoding

---

**Input:**

- Message:  $\mathbf{m} = (m_0, m_1, \dots, m_{k-1}) \in \mathbb{F}_q^{k_1}$
- Positions of partially stuck cells:  $\phi$

- 1 Compute the final message vector  $\mathbf{w} = \mathbf{m} \cdot \mathbf{G}_1$
- 2 Find  $\mathbf{z} = (z_0, z_1, \dots, z_{l-1}) \in \mathbb{F}_q$  as explained in the proof of [2, Theorem 7]
- 3 Compute the masking vector  $\mathbf{d} = \mathbf{z} \cdot \mathbf{H}_0$
- 4 Compute  $\mathbf{c} = (\mathbf{w} + \mathbf{d}) \pmod{q}$

**Output:** Codeword  $\mathbf{c} \in \mathbb{F}_q^n$

---



---

#### Algorithm 6: Decoding

---

**Input:**  $\mathbf{y} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_q^n$

- 1  $\hat{\mathbf{c}} \leftarrow$  decode  $\mathbf{y}$  in the code  $\mathcal{C}$
- 2 Unmasking Process:
  - $\hat{\mathbf{z}} \leftarrow (\hat{z}_0, \hat{z}_1, \dots, \hat{z}_{l-1})$
  - $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1, \dots, \hat{w}_{n-1}) \leftarrow (\hat{\mathbf{c}} - \hat{\mathbf{z}} \cdot \mathbf{H}_0) \pmod{q}$
  - $\hat{\mathbf{m}} \leftarrow (\hat{w}_l, \hat{w}_{l+1}, \dots, \hat{w}_{n-r-1})$

**Output:** Message vector  $\hat{\mathbf{m}} \in \mathbb{F}_q^{k_1}$

---

*Proof.* Algorithm 5 finds  $\mathbf{z}$  similar to [2, Algorithm 7] instead of only finding  $v$  value as shown in Algorithm 1. The proof follows a detailed proof in [2, Theorem 8] for the masking part which replaces the all-ones vector with an  $(n - k) \times n$  parity-check matrix. However, Theorem 4 uses  $\mathbf{H}_0$  which is  $(n - k_1 - r) \times n$ .

Since the error correction is related to  $[n, k_1]_q$  code  $\mathcal{C}_1$  and matrix  $\mathbf{G}_1$ , the proof follows Theorem 1 for how to choose  $r$  and accordingly correct  $t$  errors.  $\square$

Theorem 4 provides an extended construction of Construction 1 to mask any  $u \leq n$  partially stuck-at-1 cells and correct  $t$  errors. This gain in the number of partially stuck cells that can

Table III  
COMPARISON BETWEEN [1], [2], AND THIS WORK. NOTATION: SEE SECTION II-A.

	[1] (Stuck cells)	[2] (Partially stuck cells)	This Work (Partially stuck cells)
error correction	yes	no	yes
$k_1$	If $l = 1$ , then $k_1 = n - r - 1$	If $l = 1$ , then $k_1 \leq n - 1$	$k_1 \leq n - 1 - r$ . If $r = 0$ , it is similar to [2]
$l$	$l = n - k_1 - r$	$l = n - k_1 = 1$	$l = n - k_1 - r = 1$
$r$	$r = n - k_1 - l$	None	$r = n - k_1 - 1$
$\delta_0$ and $\delta_1$	$\delta_0$ for masking and $\delta_1$ for error correction	$\delta_0 = 2$	$\delta_0 = 2$ , $\delta_1$ follows the chosen $r \rightarrow r \leq m \lceil \frac{\delta_1 - 1}{2} \rceil$
$t$	$\leq \lfloor \frac{\delta_1 - 1}{2} \rfloor$	0	$\leq \lfloor \frac{\delta_1 - 1}{2} \rfloor$
$u$	$u < \delta_0$ and $2t < \delta_1$ , Or $u \geq \delta_0$ and $2(u + t + 1 - \delta_0) < \delta_1$	$u \leq n$ and $u < q$	$u \leq \min\{n, q - 1\}$ (Theorem 1 and 2), or $u \leq n$ (Theorem 3 and 4)
$s_{\phi_i}$	All levels $(0, \dots, q - 1)$	Partial levels $(1, \dots, q - 1)$	1 but expendable to arbitrary $s_{\phi_i}$
Name	$u$ -SMC	$u$ -PSMC	$(u, t)$ -PSMC

be masked comes at the cost of larger redundancy. However, the redundancy is still smaller than the construction for masking and error correction in [1] for the same number of (in this case stuck) cells.

## V. CONCLUSION

We have proposed several constructions for combined masking of partially stuck-at-1 cells and error correction, by combining the methods proposed in [1] for error correction and masking stuck cells, with the masking-only codes for partially stuck cells in [2]. Compared to [2], the new code constructions can correct errors in addition to masking. Furthermore, less redundancy is required for masking compared to the code constructions for stuck cells in [1].

The results can be extended to partially stuck-at- $s$  cells for  $s > 1$  using similar methods as [2, Section VII]. It is also possible to extend the cyclic construction in [1, Theorem 2] using the approach in Theorem 4, as well as the construction based on binary codes in [2, Construction 5]. Furthermore, future work should derive bounds on the required redundancy for a given number of partially stuck cells to mask and number of errors to correct.

## REFERENCES

- [1] C. Heegard, "Partitioned Linear Block Codes for Computer Memory with Stuck-at-Defects," *IEEE Transactions on Information Theory*, vol. 29, no. 6, pp. 831–842, 1983.
- [2] A. Wachter-Zeh and E. Yaakobi, "Codes for Partially Stuck-at Memory Cells," *IEEE Transactions on Information Theory*, vol. 62, no. 2, pp. 639–654, 2016.
- [3] B. Gleixner, F. Pellizzer, and R. Bez, "Reliability characterization of phase change memory," in *2009 10th Annual Non-Volatile Memory Technology Symposium (NVMTS)*. IEEE, 2009, pp. 7–11.
- [4] K. Kim and S. J. Ahn, "Reliability investigations for manufacturable high density pram," in *2005 IEEE International Reliability Physics Symposium, 2005. Proceedings. 43rd Annual*. IEEE, 2005, pp. 157–162.
- [5] S. Lee, J.-h. Jeong, T. S. Lee, W. M. Kim, and B.-k. Cheong, "A study on the failure mechanism of a phase-change memory in write/erase cycling," *IEEE Electron Device Letters*, vol. 30, no. 5, pp. 448–450, 2009.
- [6] A. Pirovano, A. Redaelli, F. Pellizzer, F. Ottogalli, M. Tosi, D. Ielmini, A. L. Lacaita, and R. Bez, "Reliability study of phase-change nonvolatile memories," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 3, pp. 422–427, 2004.
- [7] I. I. Dumer, "Asymptotically optimal codes correcting memory defects of fixed multiplicity," *Problemy Peredachi Informatsii*, vol. 25, no. 4, pp. 3–10, 1989.
- [8] —, "Asymptotically optimal linear codes correcting defects of linearly increasing multiplicity," *Problemy Peredachi Informatsii*, vol. 26, no. 2, pp. 3–17, 1990.
- [9] A. Kharazishvili and T. Tetunashvili, "Combinatorial properties of families of sets and euler-venn diagrams," *Proc. A. Razmadze Mathematical Institute*, vol. 146, pp. 117–122, 2008.

## VI. APPENDIX

*Example. Masking and Correcting Partially Stuck Memory Cells - Matrix Form* Let  $q = 3$  and  $n = 14$  and we want to store the message vector  $\mathbf{m}_1 = (0210210210210) \in \mathbb{F}_q^{n-1}$  or  $\mathbf{m}_2 = (0210210210) \in \mathbb{F}_q^{n-1-r}$  and correct one error  $t = 1$  in the tenth position. We use a ternary Hamming code with redundancy  $r = 3$ . The partially stuck positions named  $\phi_i$  are  $\phi_1 = 4$  and  $\phi_2 = 6$ ,  $\forall i \in u$  and  $u < q$  and  $u$  is the number of partially stuck cells. According to Algorithm 1, we need to find the following:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{13 \times 1} & \mathbf{I}_{13} \\ 1 & \mathbf{1}_{1 \times 13} \end{bmatrix}, \text{ and } \mathbf{P}_{(10 \times 3)} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \\ 2 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

1) *Error free* where  $r = 0$ , find  $\mathbf{w}_1 = \mathbf{m}_1 \cdot \mathbf{G}_1$ .

2) *With error*  $t = 1$ , find  $\mathbf{w}_2$ ,

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{10 \times 1} & \mathbf{I}_{10} & \mathbf{P}_{10 \times 3} \\ 1 & \mathbf{1}_{1 \times 10} & \mathbf{1}_{10 \times 3} \end{bmatrix},$$

then  $\mathbf{w}_2 = \mathbf{m}_2 \cdot \mathbf{G}_1$ .

3) Find  $\mathbf{d}$ . Since the partially stuck positions are  $w_{\phi_1} = 0$  and  $w_{\phi_2} = 1$ ,  $v \neq w_{\phi_1}$  and  $\neq w_{\phi_2}$ , then  $v = 2$ . Thus,  $z_0 = 1$ .

$$\mathbf{d} = z_0 \cdot \mathbf{G}_0 \rightarrow \mathbf{d} = 1 \cdot \mathbf{1}_{1 \times 14}$$

4) The codeword vector that can mask only is:

$$\mathbf{c} = \mathbf{w}_1 + \mathbf{d} \\ \rightarrow \mathbf{c} = 11021021021021.$$

5) The codeword vector that can mask and correct is:

$$\mathbf{c} = \mathbf{w}_2 + \mathbf{d} \rightarrow \mathbf{c} = 11021021021000$$

6) Compute  $\mathbf{s} = \mathbf{y} \cdot \mathbf{H}^T$  where  $\mathbf{H}$  is the parity check matrix of  $\mathbf{G}$ . If  $\mathbf{s} = \mathbf{0}$ , the retrieved vector  $\mathbf{y}$  is error free,  $z_0 \leftarrow c_0$ :  $\hat{\mathbf{w}}_1 = \mathbf{c} - 1 \cdot \mathbf{1}_{1 \times 14} = 00210210210210$ .

Then,  $\hat{\mathbf{m}} \leftarrow (\hat{w}_1, \dots, \hat{w}_{13})$  and we get  $\hat{\mathbf{m}} = \mathbf{m}$ .

7) To decode the retrieved vector  $\mathbf{y}$  with single error  $t = 1$  at the tenth position,  $\mathbf{y} = \mathbf{c} + \mathbf{e} = 11021021001000$ . Compute  $\mathbf{s} = \mathbf{y} \cdot \mathbf{H}^T = 110$ . It is ternary hamming code. Comparing  $\mathbf{s}^T$  to the tenth column of  $\mathbf{H}$  is to correct the error. For unmasking, same as before (repeat Step 6).