Fifty-third Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 29 - October 2, 2015

Masking Trapped Charge in Flash Memories

Antonia Wachter-Zeh and Eitan Yaakobi

Department of Computer Science
Technion—Israel Institute of Technology, Haifa, Israel
Email: $\{$*antonia, yaakobi*$\}$*@cs.technion.ac.il*

Abstract—**This paper studies defect memory cells and in particular *partially stuck-at memory cells*, which occur when charge is trapped in multi-level cells of non-volatile memories such as flash memories. If a cell can store the q levels $0, 1, \ldots, q-1$, we say that it is *partially stuck-at* level s, where $1 \leq s \leq q-1$, if it can only store values which are at least s. We follow the common setup where the encoder knows the positions and levels of the partially stuck-at cells whereas the decoder does not. In this paper, we study codes for masking u partially stuck-at cells. We derive lower and upper bounds on the redundancy of such codes and present code constructions. Furthermore, we analyze the dual defect model in which cells cannot reach higher levels, and show that codes for partially stuck-at cells can be used to mask this type of defects as well. Lastly, we analyze the capacity of the partially stuck-at memory channel and study how far our constructions are from the capacity.**

Index Terms—**(partially) stuck-at cells, flash memories**

I. Introduction

Information in flash memories is stored by electrically charging the cells with electrons in order to represent multiple levels. If charge is trapped in a cell, then its level can only be increased, or it may happen that due to defects, the cell can only represent some lower levels. Inspired by these defect models, the goal of this paper is the study of codes which mask cells that are *partially stuck-at*.

In the classical version of stuck-at cells (see e.g. [11]), the memory consists of n binary cells of which u are stuck-at either in the zero or one level. A cell is said to be *stuck-at level* $s \in \{0, 1\}$ if the value of the cell cannot be changed. Only data which matches the fixed values at the stuck-at cells can be written into the memory. A *code for stuck-at cells* maps message vectors to codewords which mask the stuck-at cells, i.e., the values of each codeword at the stuck positions coincide with their stuck level. The encoder knows the locations and values of the stuck-at cells, while the decoder does not. The task of the decoder is to reconstruct the message given only the codeword. The challenge in this defect model is to construct schemes where a large number of messages can be encoded and successfully decoded.

Codes for memories with stuck-at cells, also known as *memories with defects*, were first studied in 1974 by Kuznetsov and Tsybakov [11]. Since then, several more papers have appeared, e.g. [1]–[3], [6], [9], [10], [14], [16]–[18]. The main goal in all these works was to find constructions of codes which mask a fixed number of stuck-at cells and correct another fixed number of additional random errors. Recently, the connection between memories with stuck-at cells and the failure models of non-volatile memory cells has attracted a renewed attention to this prior work. Several more code constructions as well as efficient encoding and decoding algorithms for the earlier constructions were studied; see e.g. [4], [8], [12], [13], [15].

This paper studies codes which mask cells that are *partially stuck-at*. We assume that cells can have one of the q levels $0, 1, \ldots, q-1$. Then, it is said that a cell is *partially stuck-at* level s, where $1 \leq s \leq q-1$, if it can only store values which are at least s. We provide upper and lower bounds on the redundancy of these codes and give several constructions with small redundancy to mask partially stuck-at cells. Our constructions are first given for the case $s = 1$, and later generalized to arbitrary levels. We improve upon our constructions from [19] and analyze how far they are from our lower and upper bounds on the redundancy. In particular, if q is a multiple of $u+1$, our code construction is asymptotically optimal in terms of the redundancy. Another construction uses *binary* codes which mask (usual) stuck-at cells. We also show how the codes we propose in the paper can be used for the dual defect model in which cells cannot reach higher levels. Lastly, we analyze the capacity of the partially stuck-at memory channel and study how far our constructions are from the capacity. The results in this paper build upon our previous constructions, presented in [19].

The rest of the paper is organized as follows. In Section II, we introduce the notations and formally define the model of partially stuck-at cells studied in this paper. Our lower and upper bounds on the redundancy are presented in Section III. In Section IV, we propose codes along with encoding and decoding algorithms for the case $u < q$. In Section V we show a more general solution by using codes which mask binary stuck-at cells. Section VI generalizes the previous constructions to cells which are partially stuck-at arbitrary (possibly different) levels $s_0, s_1, \ldots, s_{u-1}$. The (dual) problem of codes for unreachable levels is studied in Section VII and Section VIII analyzes the capacity of the partially stuck-at channel. Finally, Section IX concludes this paper.

Due to space limitations, some proofs are omitted and can be found in the long version [20].

II. Definitions and Preliminaries

A. Notations

For positive integers a, b, we denote by $[a]$ the set of integers $\{0, 1, \ldots, a-1\}$ and $[a, b-1] = \{a, a+1, \ldots, b-1\}$. Vectors and matrices are denoted by lowercase and uppercase boldface letters, e.g. $\mathbf{a}$ and $\mathbf{A}$, and are indexed starting from 0. The all-zero and all-one vectors of length n are denoted by $\mathbf{0}_n$ and $\mathbf{1}_n$. Further, for a prime power q, $\mathbb{F}_q$ denotes the finite field of order q.

We consider n memory cells with q levels, i.e., they are represented as a vector in $[q]^n$. In the classical model of stuck-at cells, a cell is said to be *stuck-at level* $s \in [q]$ if it can store only the value s. In our new model of partially stuck-at cells, a cell is *partially stuck-at level* $s \in [q]$ if it can store only values which are at least s. In the first part of this work, when studying partially stuck-at cells, we only consider $s = 1$ and we call such cells *partially stuck-at*-1. Throughout this paper, we also use the notation *partially stuck-at-s*, when one or several cells are partially stuck-at level s. Lastly, in the most general case, u cells are *partially stuck-at*-$\mathbf{s}$, where $\mathbf{s} = (s_0, s_1, \dots, s_{u-1})$ is a vector and contains the different partially stuck levels $s_0, s_1, \dots, s_{u-1}$ of the u defect cells.

B. Definitions for (Partially) Stuck-at Cells

We use the notation $(n, M)_q$ to indicate a coding scheme to encode M messages into vectors of length n over the alphabet $[q]$. Its redundancy is denoted by $r = n - \log_q M$. A linear code over $[q]$ will be denoted by $[n, k]_q$, where k is its dimension and its redundancy is $r = n - k$. Whenever we speak about a linear $[n, k, d]_q$ code, then d refers to the minimum Hamming distance of an $[n, k]_q$ code. We also denote by $\rho_q(n, d)$ the smallest (known) redundancy of any linear code of length n and minimum Hamming distance d over $\mathbb{F}_q$. For the purpose of the analysis and the simplicity of notations in the paper, we let $\rho_q(n, d) = \infty$ in case q is not a power of a prime.

Codes for (partially) stuck-at cells are defined as follows.

Definition 1 (Codes for (Partially) Stuck-at Cells) *We define the following code properties:*

1) *An $(n, M)_q$ **u-stuck-at-masking code** (u-SMC) $\mathcal{C}$ is a coding scheme with encoder $\mathcal{E}$ and decoder $\mathcal{D}$. The input to the encoder $\mathcal{E}$ is the set of locations $\{\phi_0, \phi_1, \dots, \phi_{u-1}\} \subseteq [n]$, the stuck levels $s_0, s_1, \dots, s_{u-1} \in [q]$ of some $u \leq n$ stuck-at cells and a message $m \in [M]$. Its output is a vector $\mathbf{y}^{(m)} \in [q]^n$ which matches the values of the u stuck-at cells, i.e.,*
$$y^{(m)}_{\phi_i} = s_i, \ \forall i \in [u],$$
and its decoded value is m, that is $\mathcal{D}(\mathbf{y}^{(m)}) = m$.

2) *An $(n, M)_q$ **$(u, \mathbf{s})$-partially-stuck-at-masking code** ($(u, \mathbf{s})$-PSMC) $\mathcal{C}$ is a coding scheme with encoder $\mathcal{E}$ and decoder $\mathcal{D}$. The input to the encoder $\mathcal{E}$ is the set of locations $\{\phi_0, \phi_1, \dots, \phi_{u-1}\} \subseteq [n]$, the partially stuck levels $\mathbf{s} = (s_0, s_1, \dots, s_{u-1}) \in [1, q-1]^u$ of some $u \leq n$ partially stuck-at cells and a message $m \in [M]$. Its output is a vector $\mathbf{y}^{(m)} \in [q]^n$ which masks the values of the u partially stuck-at cells, i.e.,*
$$y^{(m)}_{\phi_i} \geq s_i, \ \forall i \in [u],$$
and its decoded value is m, that is $\mathcal{D}(\mathbf{y}^{(m)}) = m$.

Notice that in contrast to classical error-correcting codes, (P)SMCs are not just a set of codewords, but also an explicit coding scheme with encoder and decoder. The partially stuck levels stored in $\mathbf{s}$ have to be known to the decoder of a PSMC, whereas the decoders of SMCs are independent of the stuck levels. Clearly, a $(u, \mathbf{s})$-PSMC is also a $(u', \mathbf{s}')$-PSMC, where $u' \leq u$ and $\mathbf{s}'$ is a subvector of $\mathbf{s}$ of length u'.

Whenever we speak about (u, s)-PSMCs (i.e., s is a scalar), we refer to a code that masks at most u partially-stuck-at-s cells and when we speak about a u-PSMC, we mean a code which masks at most u partially-stuck-at-1 cells.

Our goal is to minimize the redundancy $n - \log_q M$ for fixed values u and $\mathbf{s}$, while providing efficient encoding and decoding algorithms. For positive integers n, q, u, where $u \leq n$, and a vector $\mathbf{s} \in [1, q-1]^u$, we denote by $r_q(n, u, \mathbf{s})$ the *minimum redundancy* of a $(u, \mathbf{s})$-PSMC over $[q]$.

C. Stuck-At-Masking Codes

Theorem 1 (Masking Stuck-At Cells, [6]) *Let $\mathcal{C}$ be an $[n, k, d]_q$ code with $d \geq u + 1$, where q is a prime power. Then, there exists a u-SMC with redundancy $r = n - k$.*

It is not completely known whether the scheme of Theorem 1 is optimal with respect to its achieved redundancy. However, the redundancy of such a code has to be at least u since there are u stuck-at cells that cannot store information.

III. Bounds on the Redundancy

For deriving upper and lower bounds on the minimum redundancy of PSMCs, we assume the most general case of $(u, \mathbf{s})$-PSMCs, where $\mathbf{s} = (s_0, s_1, \dots, s_{u-1}) \in [1, q-1]^u$.

Theorem 2 (Bounds on the Redundancy) *For any number of $u \leq n$ partially stuck-at cells and any levels $\mathbf{s} = (s_0, s_1, \dots, s_{u-1}) \in [1, q-1]^u$, the value of the minimum redundancy $r_q(n, u, \mathbf{s})$ to mask these cells satisfies*
$$\begin{aligned} u - \log_q \left(\prod_{i=0}^{u-1} (q - s_i) \right) \\ \leq r_q(n, u, \mathbf{s}) \\ \leq \min \left\{ n \cdot \left(1 - \log_q \left(q - \max_i\{s_i\}\right)\right), \ \rho_q(n, u+1) \right\}. \end{aligned}$$

Proof: First, we prove the lower bound. The $n - u$ cells, which are not partially stuck-at, can each carry a q-ary information symbol and the u partially stuck-at cells can still represent $q - s_i$ possible values (all values except for $[s_i]$). Hence, we can store at most $M \leq q^{n-u} \prod_{i=0}^{u-1}(q - s_i)$ q-ary vectors and the code redundancy satisfies
$$r \geq n - \log_q \left(q^{n-u} \prod_{i=0}^{u-1} (q - s_i) \right) = u - \log_q \left(\prod_{i=0}^{u-1} (q - s_i) \right).$$

Second, we prove the upper bound. A trivial construction to mask any $u \leq n$ partially stuck-at cells is to use only the values $\max_i\{s_i\}, \dots, q-1$ as possible symbols in any cell. Any cell therefore stores $\log_q(q - \max_i\{s_i\})$ q-ary information symbols. The achieved redundancy is

$$n - \log_q\big((q - \max_i\{s_i\})^n\big) = n\big(1 - \log_q(q - \max_i\{s_i\})\big),$$

and therefore $r_q(n, u, \mathbf{s}) \leq n\big(1 - \log_q(q - \max_i\{s_i\})\big)$.

Furthermore, every u-SMC can also be used as $(u, \mathbf{s})$-PSMC since the SMC restricts the values of the stuck-at cells more than the PSMC. With Theorem 1, the redundancy of an SMC is $\rho_q(n, u+1)$, which provides another upper bound on the value of $r_q(n, u, \mathbf{s})$. ■

Note that $\rho_q(n, u+1) \geq u$ (by the Singleton bound).

The bounds from Theorem 2 are a generalization of the bounds from [20, Equation (1)] which were proven for the case $s = 1$ only. For partially stuck-at-s cells, we further improve the lower bound on the redundancy in the next theorem. This bound is new and did not appear in [19].

Theorem 3 (Improved Lower Bound) *For any $(n, M)_q$ (u, s)-PSMC, we have $M \leq \lfloor \frac{q^n + u(q-s)^n}{u+1} \rfloor$, and therefore*

$$r_q(n, u, s) \geq \log_q(u+1) - \log_q\big(1 + u(1 - s/q)^n\big).$$

Proof: Assume that there exists an $(n, M)_q$ (u, s)-PSMC, and let $\mathcal{E}$, $\mathcal{D}$ be its encoder, decoder, respectively. In this case we assume that the encoder's input is a message $m \in [M]$ and a set of indices $U \subseteq [n], |U| \leq u$, of the locations of the cells which are partially stuck-at level s. We will show that $M \leq \lfloor \frac{q^n + (q-s)^n}{u+1} \rfloor$. For every $m \in [M]$, let

$$\mathcal{D}^{-1}(m) = \{\mathbf{y} \in [q]^n \mid \mathcal{D}(\mathbf{y}) = m\}.$$

For every $m \in [M]$ such that $\mathcal{D}^{-1}(m) \cap ([q] \setminus [s])^n = \emptyset$, we have that $|\mathcal{D}^{-1}(m)| \geq u+1$. To see that, assume on the contrary that $|\mathcal{D}^{-1}(m)| = u$ (the same proof holds if $|\mathcal{D}^{-1}(m)| < u$), so we can write $\mathcal{D}^{-1}(m) = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{u-1}\}$, while $\mathbf{y}_j \notin ([q] \setminus [s])^n$ for $j \in [u]$. For $j \in [u]$, let $i_j \in [n]$ be such that $y_{j,i_j} < s$, and $U = \{i_0, i_1, \ldots, i_{u-1}\}$. Then, $\mathcal{E}(m, U) \neq \mathbf{y}_j$ for all $j \in [u]$ (where the message m and the positions set U are the input to the encoder $\mathcal{E}$) and thus $|\mathcal{D}^{-1}(m)| \geq u+1$, in contradiction.

Since there are $(q-s)^n$ vectors all with values at least s, there are $M - (q-s)^n$ vectors of the described type above where for each such vector there exists a message m where $|\mathcal{D}^{-1}(m)| \geq u+1$. Therefore, we get that $(q-s)^n + (u+1)(M - (q-s)^n) \leq q^n$, or

$$M \leq \left\lfloor \frac{q^n + u(q-s)^n}{u+1} \right\rfloor.$$

We also conclude that

$$r_q(n, u, s) \geq \log_q(u+1) - \log_q\big(1 + u(1 - s/q)^n\big). \quad ■$$

IV. A Construction for Partially Stuck-at Level $s = 1$ Cells for $u < q$

In [19, Section III], we have presented a construction for masking $u < q$ partially stuck-at memory cells. For all n and any $u < q$, this construction provides a PSMC with redundancy of one symbol. In this section, we use a similar principle as in [19, Section III] but reduce the redundancy. Furthermore, we will show that if q is a multiple of $u+1$, our new construction is asymptotically optimal.

Theorem 4 (Construction A) *If $u < q$ and $u \leq n$, then for all n, there exists a u-PSMC over $[q]$ of length n and redundancy*

$$r = 1 - \log_q \left\lfloor \frac{q}{u+1} \right\rfloor.$$

Proof: In order to prove the statement, we show the encoding and decoding procedures for Construction A.

Algorithm 1. ENCODING-A$(\mathbf{m}; m'; \phi_0, \phi_1, \ldots, \phi_{u-1})$

```
Input: • message: $\mathbf{m} = (m_0, m_1, \ldots, m_{n-2}) \in [q]^{n-1}$
       • additional message: $m' \in \left[\lfloor \frac{q}{u+1} \rfloor\right]$
       • positions of stuck-at-1 cells:
         $\{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n]$
1 $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1}) \leftarrow (0, m_0, m_1, \ldots, m_{n-2})$
2 Set $v \in [u+1]$ such that
  $v \notin \{w_{\phi_0} \bmod (u+1), \ldots, w_{\phi_{u-1}} \bmod (u+1)\}$
3 $z \leftarrow q - v - m'(u+1)$
4 $\mathbf{y} \leftarrow (\mathbf{w} + z \cdot \mathbf{1}_n) \bmod q$
Output: vector $\mathbf{y}$ with $y_{\phi_i} \geq 1, \forall i \in [u]$
```

We will prove that the output vector $\mathbf{y}$ masks the partially stuck-at-1 cells. In Step 2 of Algorithm 1, we can always find a value $v \in [u+1]$ as required since the set $\{w_{\phi_0} \bmod (u+1), \ldots, w_{\phi_{u-1}} \bmod (u+1)\}$ has cardinality u and there are $u+1$ possible values to choose from. Note that in Step 3, $z \in [q]$ since $v \in [u+1]$ and $m'(u+1) \in [q-u]$. In Step 4, we obtain:

$$y_{\phi_i} = (w_{\phi_i} - v - m'(u+1)) \bmod q.$$

Since $v \neq w_{\phi_i} \bmod (u+1)$ from Step 2, $y_{\phi_i} \neq 0$, for all $i \in [u]$, and the partially stuck-at-1 positions are masked.

Algorithm 2. DECODING-A$(\mathbf{y})$

```
Input: • stored vector: $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in [q]^n$
1 $\widehat{z} \leftarrow y_0$
2 $\widehat{v} \leftarrow (q - \widehat{z}) \bmod (u+1)$
3 $\widehat{m'} \leftarrow \frac{q - \widehat{z} - \widehat{v}}{u+1}$
4 $\widehat{\mathbf{m}} \leftarrow ((y_1, y_2, \ldots, y_{n-1}) - \widehat{z} \cdot \mathbf{1}_{n-1}) \bmod q$
Output: message vector $\widehat{\mathbf{m}} \in [q]^{n-1}$
        additional message $\widehat{m'}$
```

The decoding algorithm has to guarantee that we can recover $\mathbf{m}$ and m' and is shown in Algorithm 2. Here, we notice that $y_0 = z = q - v - m'(u+1)$ from the encoding step and therefore, in Step 1, $\widehat{z} = z$ and in Step 2, $\widehat{v} = v$. Solving $y_0 = q - v - m'(u+1)$ for m' shows that in Step 3, the result of the fraction is always an integer and that $\widehat{m'} = m'$. Since $\widehat{z} = z$, it follows that $\widehat{\mathbf{m}} = \mathbf{m}$ in Step 4.

The statement on the redundancy follows since we have one redundancy symbol (the first position), but the additional information m' is stored there, which reduces the redundancy by $\log_q \lfloor \frac{q}{u+1} \rfloor$ in terms of q-ary symbols. ■

The difference to [19, Section III] is the additional information m' that can be stored with Construction A. The principle is also illustrated in the following example.

Example 1 *Let $u = 2$, $q = 6$ and $n = 5$. We choose $\mathbf{m} = (0, 1, 5, 2, 4)$ and $m' = 1 \in [2]$. Assume $\phi_0 = 1$ and $\phi_1 = 5$. We follow the steps of Algorithm 1 and obtain $\mathbf{w} = (0, 0, 1, 5, 2, 4)$. In Step 2, v should not be in $\{0, 1\}$ but in $[3]$ and therefore $v = 2$. In Step 3, we obtain $z = 1$ and finally, Step 4 gives $\mathbf{y} = (1, 1, 2, 0, 3, 5)$ and we see that $y_{\phi_0} = y_1 = 1$ and $y_{\phi_1} = y_5 = 5$ and the partially stuck-at cells are masked.*

The decoder from Algorithm 2 knows $\mathbf{y}$ and calculates in Step 1: $\widehat{z} = 1$, in Step 2: $\widehat{v} = 2$ and in Step 3: $m' = 1$. Finally, we can correctly reconstruct $\widehat{\mathbf{m}} = (0, 1, 5, 2, 4)$.

The important difference to [19, Section III] is that z can always be chosen from a small subset of $[q]$, namely from $[u+1]$ (whereas in [19, Section III], z was chosen from $[q]$). This makes it possible to store the "additional information" of m' in the first cell. Thus, the information stored (in terms of q-ary symbols) increases by $\log_6(6/3)$ and the required redundancy reduces from one symbol to $1 - \log_6(2) \approx 0.613$ symbols while the lower bound from Theorem 2 gives ≈ 0.204. The lower bound from Theorem 3 gives 0.284 for $n = 5$. For $n \to \infty$, the lower bound from Theorem 3 reaches 0.613 which is exactly the redundancy achieved by our construction.

Theorem 5 (Optimality of Theorem 4) *If $(u + 1)$ divides q, then the u-PSMC from Theorem 4 is asymptotically optimal in terms of the redundancy.*

Proof: If $(u+1)|q$, then the redundancy from Theorem 4 is $r = \log_q(u + 1)$. The lower bound on the minimum redundancy from Theorem 3 is $r_q(n, u, \mathbf{1}) = \log_q(u + 1) - \log_q\big(1 + u(1 - 1/q)^n\big)$. The difference between both is

$$\Delta_q(n, u) = r - r_q(n, u, \mathbf{1}) = \log_q\big(1 + u(1 - 1/q)^n\big).$$

For $n \to \infty$ and since $(1 - 1/q) < 1$, this value approaches

$$\log_q\big(1 + u(1 - 1/q)^n\big) \to \log_q\big(1 + 0\big) = 0.$$

Thus, the construction is asymptotically optimal. ■

V. Construction Using Binary Codes for Partially Stuck-at Level $s = 1$ Cells

In this section, we describe a construction of u-PSMCs for masking u partially stuck-at-1 cells by means of *binary* SMCs. This construction works for any u; however, for $u < q$, the construction from the previous section achieves smaller redundancy, therefore and unless stated otherwise, we assume in this section that $n \geq u \geq q$.

For a vector $\mathbf{w} \in [q]^n$ and a set $U \subseteq [n]$, the notation $\mathbf{w}_U$ denotes the subvector of $\mathbf{w}$ of length $|U|$ which consists of the positions in U. Let U contain the locations of the u partially stuck-at-1 cells, and let a vector $\mathbf{w} \in [q]^n$ be given, then our construction of PSMCs has two main parts:

1) Find $z \in [q]$ such that the number of zeros and $(q-1)$s is minimized in the vector
$$(\mathbf{w}^{(z)})_U = (\mathbf{w} + z \cdot \mathbf{1}_{n+1})_U,$$
and denote this value by $\widetilde{u}$.
2) Use a binary $\widetilde{u}$-SMC to mask these $\widetilde{u}$ stuck-at cells.

This idea provides the following construction. Later, we will demonstrate this idea with two examples.

Theorem 6 (Construction B) *Let n, $q \geq 4$ and $u \leq n$ be positive integers and define $\widetilde{u} = \lfloor 2u/q \rfloor$. Assume that $\widetilde{\mathcal{C}}$ is an $[n, k, d \geq \widetilde{u}+1]_2$ binary $\widetilde{u}$-SMC with encoder $\widetilde{\mathcal{E}}$ and decoder $\widetilde{\mathcal{D}}$ given by Theorem 1. Then, there exists a u-PSMC over $[q]$ of length $(n + 1)$ and redundancy*

$$r = (n - k - 1) \log_q \left(\frac{q}{\lfloor q/2 \rfloor} \right) + 2.$$

Proof: We will give the explicit algorithms for encoding and decoding. We assume that $\mathbf{H}$ is a systematic $(n-k) \times n$ parity-check matrix of the code $\widetilde{\mathcal{C}}$ and we refer to Theorem 1 as the encoder $\widetilde{\mathcal{E}}$ and decoder $\widetilde{\mathcal{D}}$ of the code $\widetilde{\mathcal{C}}$, respectively.

Let us start with the encoding algorithm for the u-PSMC.

```
Algorithm 3. ENCODING-B$(\mathbf{m}; \mathbf{m}'; U)$

Input: • messages: $\mathbf{m} = (m_0, m_1, \dots, m_{k-1}) \in [q]^k$
         and
         $\mathbf{m}' = (m'_0, m'_1, \dots, m'_{n-k-2}) \in \big[\lfloor q/2 \rfloor\big]^{n-k-1}$
       • positions of partially stuck-at-1 cells:
         $U = \{\phi_0, \phi_1, \dots, \phi_{u-1}\} \subseteq [n+1]$
1 $\mathbf{w} = (w_0, w_1, \dots, w_n) \leftarrow (\mathbf{0}_{n-k}, m_0, m_1, \dots, m_{k-1}, 0)$
2 Find $z \in [q]$ such that the vector $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$
  zeros and $(q-1)$s, where $\mathbf{w}^{(z)} = (\mathbf{w} + z \cdot \mathbf{1}_{n+1}) \bmod q$.
  Let $\widetilde{U}$ be the set of these positions.
3 If $z > 0$ then $w_n^{(z)} \leftarrow z$, else $w_n^{(z)} \leftarrow q - 2$
4 Let $\mathbf{v} = (v_0, v_1, \dots, v_{n-1}) \in [2]^n$ be such that $v_i \leftarrow 1$ if
  $w_i^{(z)} = q - 1$, else $v_i \leftarrow 0$
5 Apply Theorem 1:
  $\mathbf{c}' \leftarrow \widetilde{\mathcal{E}}\big((v_{n-k}, v_{n-k+1}, \dots, v_{n-1}); \widetilde{U}; \mathbf{1}_n\big)$
  $\widetilde{\mathbf{c}} \leftarrow (\mathbf{c}'\ 0)$
6 $\widetilde{\mathbf{m}} \leftarrow (2m'_0, 2m'_1, \dots, 2m'_{n-k-2}, \mathbf{0}_{k+2}) \in [q]^{n+1}$
7 $\mathbf{y} \leftarrow (\mathbf{w}^{(z)} + \widetilde{\mathbf{c}} + \widetilde{\mathbf{m}}) \bmod q$

Output: vector $\mathbf{y} \in [q]^{n+1}$ with $y_{\phi_i} \geq 1$, $\forall i \in [u]$
```

First, we show that in Step 2 of Algorithm 3, there exists $z \in [q]$ such that the vector $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$ zeros and $(q-1)$s, where $\mathbf{w}^{(z)} = (\mathbf{w} + z \cdot \mathbf{1}) \bmod q$. The value $\widetilde{u}$ is equivalent to the minimum number of cells in any two (cyclically) consecutive levels in $\mathbf{w}$. That is, let us denote $v_i = |\{j : w_j = i, j \in \{\phi_0, \dots, \phi_{u-1}\}|$, for all $i \in [q]$. Then, we seek to minimize the value of

$$v_i + v_{(i+1) \bmod q},$$

where $i \in [q]$. Since $\sum_{i=0}^{q-1}(v_i + v_{(i+1) \bmod q}) = 2u$, with the pigeonhole principle, there exists an integer $z \in [q]$ such that $v_z + v_{(z+1) \bmod q} \leq \lfloor 2u/q \rfloor = \widetilde{u}$. Therefore, the vector $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$ zeros and $(q-1)$s, and the set of these positions is denoted by $\widetilde{U}$. We write the value of z in the last cell and if $z = 0$, we write $w_n = q-1$. This is necessary in the case where the last cell is partially stuck-at-1.

Next, we treat the $\widetilde{u}$ cells of $\mathbf{w}^{(z)}$ of value 0 or $q-1$ as binary stuck-at cells since adding 0 or 1 to the other $u - \widetilde{u}$ cells still masks those partially stuck-at-1 cells as they will not reach level 0. Therefore, according to Theorem 1, we can use the encoder of $\widetilde{\mathcal{E}}$ of the code $\widetilde{\mathcal{C}}$ as done in Steps 4 and 5. The first part of Step 5 is a call of Theorem 1. In order to do so, we first generate a length-n binary vector $\mathbf{v}$ where its value is 1 if and only if the corresponding cell has value $q-1$. Then, we require that the $\widetilde{u}$ cells will be stuck-at level 1. This will guarantee that the output $\widetilde{\mathbf{c}}$ has value 1 if the corresponding partially stuck-at cell was in level 0 and its value is 0 if the corresponding cell was in level $q-1$.

The first $n-k$ cells contain so far only binary values, thus, we use the first $n-k-1$ cells to write another symbol with $\lfloor q/2 \rfloor$ values in each cell, as done in Steps 6 and 7. The last cell in this group of $n-k$ cells remains to store only a binary value, which is necessary to determine the value of z in the decoding algorithm.

To complete the proof, we present the decoding algorithm.

Algorithm 4. DECODING-B$(\mathbf{y})$

```
Input: • stored vector: $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in [q]^{n+1}$
1 If $(y_{n-k-1} - y_n) \bmod q \leq 1$ then $\widehat{z} \leftarrow y_n$, else $\widehat{z} \leftarrow 0$
2 $\widehat{\mathbf{y}} \leftarrow \mathbf{y} - \widehat{z} \cdot \mathbf{1}_{n+1}$
3 $\widehat{\mathbf{m}'} \leftarrow (\lfloor \widehat{y}_0/2 \rfloor, \lfloor \widehat{y}_0/2 \rfloor, \ldots, \lfloor \widehat{y}_{n-k-2}/2 \rfloor)$
4 $\widehat{\mathbf{t}} \leftarrow (\widehat{y}_0 - 2\widehat{m}'_0, \ldots, \widehat{y}_{n-k-2} - 2\widehat{m}'_{n-k-2}, \widehat{y}_{n-k-1}) \bmod q$
5 $\widehat{\mathbf{c}'} \leftarrow \widehat{\mathbf{t}} \cdot \mathbf{H}$
6 $\widehat{\mathbf{m}} \leftarrow (\widehat{y}_{n-k} - \widehat{c}'_{n-k}, \ldots, \widehat{y}_{n-1} - \widehat{c}'_{n-1}) \bmod q$
Output: message vectors $\widehat{\mathbf{m}} \in [q]^k$ and
        $\widehat{\mathbf{m}'} \in [\lfloor q/2 \rfloor]^{n-k-1}$
```

In Step 1, we first determine the value of z. Note that if $z \neq 0$ then $y_{n-k} = y_n = z$ or $y_{n-k} = (y_n + 1) \bmod q$ and in either case the condition in this step holds. In case $z = 0$ then $y_n = q-2$ and $y_{n-k} = 0$ or $y_{n-k} = 1$ so $(y_{n-k-1} - y_n) \bmod q \geq 2$ (since $q \geq 4$). Therefore, we conclude that $\widehat{z} = z$.

In Step 3, we determine the value of the second message $\mathbf{m}'$. Since $\widehat{z} = z$, we get that $\widehat{\mathbf{m}'} = \mathbf{m}'$. In Steps 4 and 5, we follow the principle behind Theorem 1 in order to decode the vector $\mathbf{c}'$ and get that $\widehat{\mathbf{c}'} = \mathbf{c}'$. Lastly, in Step 5 we retrieve the first message vector and get $\widehat{\mathbf{m}} = \mathbf{m}$.

The size of the message we store in this code is $M = q^k \cdot \lfloor (q/2) \rfloor^{n-k-1}$ and the redundancy of this u-PSMC is

$$r = (n-k-1)\log_q\left(\frac{q}{\lfloor q/2 \rfloor}\right) + 2.$$

■

The following examples show the complete encoding and decoding processes described in Theorem 6.

Example 2 *Let $q = 4$, $n = 15$, $k = 11$, $u = 5$ and $U = \{1, 4, 8, 12, 15\} \subset [16]$ be the set of partially stuck-at-1 positions. Then, $\widetilde{u} = \lfloor 2u/q \rfloor = 2$ and we can use the $[15, 11, 3]_2$ code $\widetilde{\mathcal{C}}$ as the binary 2-SMC. The following matrix is a systematic parity-check matrix of $\widetilde{\mathcal{C}}$:*

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Let $\mathbf{m} = (0, 3, 2, 1, 2, 2, 3, 1, 3, 2, 2)$ and $\mathbf{m}' = (1, 0, 1)$.

Let us first show the steps of the encoding algorithm. With $z = 1$ in Step 2, we have in the different steps of Algorithm 3 (the partially stuck-at-1 positions are underlined):

Step 1: $\mathbf{w} = (0, \underline{0}, 0, 0, \underline{0}, 3, 2, 1, \underline{2}, 2, 3, 1, \underline{3}, 2, 2, \underline{0})$

Steps 2&3: $\mathbf{w}^{(z)} = (1, 1, 1, 1, 1, 0, 3, 2, 3, 3, 0, 2, 0, 3, 3, 1)$

Therefore, $(\mathbf{w}^{(z)})_U = (1, 1, 3, 0, 1)$ has two cells in level 0 or 3 which equals $\widetilde{u}$. Further, we obtain:

Step 4: $\mathbf{v} = (0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1)$

Step 5: $\mathbf{c}' = (1, 0, 0, 0) \cdot \mathbf{H} =$
$= (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)$
$\widetilde{\mathbf{c}} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0)$

Step 6: $\widetilde{\mathbf{m}} = (2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

Step 7: $\mathbf{y} = (0, \underline{1}, 3, 1, \underline{1}, 0, 3, 2, \underline{3}, 0, 1, 3, \underline{1}, 0, 0, \underline{1})$

Clearly, the partially stuck-at-1 positions are masked in the vector $\mathbf{y}$.

Let us now show the decoding process, i.e., Algorithm 4.

Step 1: $y_3 - y_{15} = 0 \Longrightarrow \widehat{z} = 1$

Step 2: $\widehat{\mathbf{y}} = (3, 0, 2, 0, 0, 3, 2, 1, 2, 3, 0, 2, 0, 3, 3, 0)$

Step 3: $\widehat{\mathbf{m}'} = (\lfloor 3/2 \rfloor, 0, \lfloor 2/2 \rfloor) = (1, 0, 1)$

Step 4: $\widehat{\mathbf{t}} = (3-2, 0-0, 2-2, 0) = (1, 0, 0, 0)$

Step 5: $\widehat{\mathbf{c}'} = \widehat{\mathbf{t}} \cdot \mathbf{H} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1)$

Step 6: $\widehat{\mathbf{m}} = (0, 3, 2, 1, 2, 2, 3, 1, 3, 2, 2)$.

We have therefore successfully recovered the messages $\mathbf{m}$ and $\mathbf{m}'$ and the redundancy to mask these five partially stuck-at-1 cells is $r = 3.5$ q-ary cells.

The lower bound from Theorem 2 gives 1.037 and the lower bound from Theorem 3 gives 1.26.

As a comparison, to mask $u = 5$ usual stuck-at cells in a block of 16 cells, we need a quaternary code of length $n = 16$ and distance $d \geq u + 1 = 6$. The largest such code is a $[16, 9, 6]_4$ code with 7 redundancy symbols. The trivial construction from Theorem 2 needs redundancy 3.11. For this example, the new construction is thus worse than the trivial one. However, for larger n, the influence of the $+2$ in the redundancy diminishes and our construction outperforms the trivial one (see Example 3), but the principle of the construction is easier to show with short vectors.

Example 3 *Let $q = 4$, $n = 63$, $k = 57$ and $u = 5$. Then, the required redundancy for Construction B from Theorem 6 is $r = 4.5$. This improves significantly upon the upper bounds from Theorem 2 since the trivial construction from Theorem 2 needs redundancy 13.1 and $\rho_4(64, 6) = 13$.*

The next corollary summarizes this upper bound on the redundancy for n cells.

Corollary 1 *For all $4 \leq q \leq u \leq n$ we have that*

$$r_q(n, u, \mathbf{1}) \leq \left(\rho_2\left(n-1, \lfloor \tfrac{2u}{q} \rfloor + 1\right) - 1\right) \cdot \log_q\left(\frac{q}{\lfloor q/2 \rfloor}\right) + 2.$$

Note that for $u \to n$ and for large q, the trivial construction from Section II is quite good. Construction B outperforms the trivial construction if $u > q$ and $n \gg u$.

VI. Generalization of the Constructions to Arbitrary Partially Stuck-at Levels

The main goal of this section is to consider the generalized model of partially stuck-at cells as in Definition 1. This model is applicable in particular for non-volatile memories where the different cells might be partially stuck-at different levels.

Let $U = \{\phi_0, \phi_1, \ldots, \phi_{u-1}\}$ be the set of locations of the partially stuck-at cells where the i-th cell, for $i \in \{\phi_0, \phi_1, \ldots, \phi_{u-1}\}$, is partially stuck-at level s_i. We denote by u_i, $i \in [q]$, the number of cells which are partially stuck-at level i (thus, $\sum_{i=1}^{q-1} u_i = u$) and by $U_i \subseteq [n]$, $\forall i \in [q]$, the positions of the cells which are partially stuck-at-i.

A. Generalization of Construction A

Theorem 7 (Generalized Construction A) *If $\sum_{i=0}^{u-1} s_i < q$ and $u \leq n$, then for all n, there exists a $(u, \mathbf{s})$-PSMC, where $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [0, q-1]^u$, over $[q]$ of length n and redundancy*

$$r = 1 - \log_q \left\lfloor \frac{q}{\sum_{i=0}^{u-1} s_i + 1} \right\rfloor.$$

The encoding and decoding processes are analogous to Algorithms 1 and 2, respectively. Unfortunately, we cannot claim (as in Theorem 5 for $s_i = 1$, $\forall i$) that this construction is asymptotically optimal, since the lower bound on the redundancy in Theorem 3 does not depend asymptotically on the value of s.

B. Generalization of Construction B

Let us now describe a method to generalize the construction from Theorem 6 to the case where the cells are partially stuck-at different levels, given by $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1})$. We want to use a Q-ary SMC, where $Q \geq \max_i\{s_i\}+1$, to obtain an $(u, \mathbf{s})$-PSMC. We do this by refining the "minimization step" as done in the following theorem.

Theorem 8 (Generalized Construction B) *Let n, $q \geq 4$ and u be positive integers. Let $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$ be given and denote $u_i = |\{s_j = i, \forall j \in [u]\}|$, $\forall i \in [q]$. Then, $u = \sum_{i=1}^{q-1} u_i$ and let $Q \geq \max_i\{s_i\} + 1$ be a prime power. Denote $\sigma_i = \min\{q, Q + i - 1\}$, $\forall i \in [q]$, and*

$$\widetilde{u} = \left\lfloor \frac{\sum_{i=1}^{q-1} u_i \cdot \sigma_i}{q} \right\rfloor.$$

Assume that $\widetilde{\mathcal{C}}$ is an $[n, k, d \geq \widetilde{u}+1]_Q$ Q-ary $\widetilde{u}$-SMC with encoder $\widetilde{\mathcal{E}}$ and decoder $\widetilde{\mathcal{D}}$ given by Theorem 1. Then, there exists a $(u, \mathbf{s})$-PSMC over $[q]$ of length $(n+1)$ and redundancy

$$r = (n - k - 1) \log_q \left(\frac{q}{\lfloor q/Q \rfloor}\right) + 2. \qquad (1)$$

Proof: If a cell in U_i has a cell level in $[i, q-Q]$, then this partially stuck-at cell is still masked after adding any value from $[Q]$. Thus, we want to find $z \in [q]$ such that the partially stuck-at positions in the vector $(\mathbf{w}^{(z)})_{U_i} = (\mathbf{w} + z \cdot \mathbf{1}_{n+1})_{U_i}$ contain as many values in $[i, q-Q]$, $\forall i \in [1, q-1]$, as possible. Equivalently, we want to minimize the number of values from $[q] \setminus [i, q-Q]$ in $(\mathbf{w}^{(z)})_{U_i}$, for all $i \in [1, q-1]$.

Let us describe a way how accomplish this. We have

$$\sigma_i \overset{\text{def}}{=} \big|[q] \setminus [i, q-Q]\big| = \min\{q, Q + i - 1\}, \ \forall i \in [q],$$

and generalizing Theorem 6, we define:

$$v_\ell^{(i)} = |\{j : w_j = \ell, j \in U_i\}|, \ \forall \ell \in [q], \ i \in [1, q-1]. \quad (2)$$

We want to minimize (subject to $\ell \in [q]$)

$$\sum_{i=1}^{q-1} \sum_{j=0}^{\sigma_i - 1} v_{\ell+j \bmod q}^{(i)}.$$

We know that $\sum_{\ell=0}^{q-1} \sum_{i=1}^{q-1} \sum_{j=0}^{\sigma_i-1} v_{\ell+j \bmod q}^{(i)} = \sum_{i=1}^{q-1} u_i \cdot \sigma_i$. Thus, by the pigeonhole principle, there exists an integer $\ell \in [q]$ such that

$$\sum_{i=1}^{q-1} \sum_{j=0}^{\sigma_i - 1} v_{\ell+j \bmod q}^{(i)} \leq \left\lfloor \frac{\sum_{i=1}^{q-1} u_i \cdot \sigma_i}{q} \right\rfloor \overset{\text{def}}{=} \widetilde{u}.$$

Similarly to Theorem 6, we can treat these $\widetilde{u}$ partially stuck-at cells as Q-ary cells which are stuck-at. A similar proof as in Theorem 6 leads to the statement. ■

VII. Codes for Cells with Unreachable Levels

In flash memories, it might happen that certain levels cannot be reached or should not be programmed anymore since they are highly unreliable, see e.g. [5]. Finding codes that mask these cells can be seen as the dual problem to finding PSMCs. Namely, we want to find a code as follows.

Definition 2 (Codes for Unreachable Levels) *An $(n, M)_q$ $(u, \mathbf{s})$-**unreachable-masking code** ($(u, \mathbf{s})$-UMC) $\mathcal{C}$ is a coding scheme with encoder $\mathcal{E}$ and decoder $\mathcal{D}$. The input to the encoder $\mathcal{E}$ is the set of locations $\{\phi_0, \ldots, \phi_{u-1}\} \subseteq [n]$, the unreachable levels $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [q-1]^u$ of some $u \leq n$ cells and a message $m \in [M]$. Its output is* *a vector $\mathbf{y}^{(m)} \in [q]^n$ which masks the values of the u cells with unreachable levels, i.e.,*

$$y_{\phi_i}^{(m)} \leq s_i, \ \forall i \in [u],$$

and its decoded value is m, that is $\mathcal{D}(\mathbf{y}^{(m)}) = m$.

Theorem 9 *Assume there exists a $(u, \mathbf{s})$-PSMC with $\mathbf{s} = (q-1-s_0, q-1-s_1, \ldots, q-1-s_{u-1})$ and redundancy r. Then, this code can be used as a $(u, \widetilde{\mathbf{s}})$-UMC with $\widetilde{\mathbf{s}} = (s_0, s_1, \ldots, s_{u-1})$ and redundancy r.*

Proof: Assume we want to write a message into n memory cells where the levels at positions $\phi_0, \phi_1, \ldots, \phi_{u-1}$ can be at most $s_0, s_1, \ldots, s_{u-1}$. We construct a $(u, \mathbf{s})$-PSMC with redundancy r for the levels $\mathbf{s} = (q-1-s_0, q-1-s_1, \ldots, q-1-s_{u-1})$. The output of this encoder is a vector $\mathbf{y}^{(m)}$ such that $y_{\phi_i}^{(m)} \geq q-1-s_i, \ \forall i \in [0, u-1]$.

Therefore, the vector $\overline{\mathbf{y}^{(m)}} \overset{\text{def}}{=} (q-1-y_0^{(m)}, q-1-y_1^{(m)}, \ldots, q-1-y_{n-1}^{(m)})$ has the property that

$$\overline{y_{\phi_i}^{(m)}} \leq q-1-(q-1-s_i) = s_i, \quad \forall i \in [0, u-1],$$

and therefore, we write $\overline{\mathbf{y}^{(m)}}$ into the cells and have constructed a $(u, \widetilde{\mathbf{s}})$-UMC with $\widetilde{\mathbf{s}} = (s_0, s_1, \ldots, s_{u-1})$. ■

It is easy to show that our constructions improve on both, [5] and [6], when used for masking memory cells with unreachable levels.

VIII. The Capacity of the Partially Stuck-at Cell Channel

In this section, we study the setup of partially stuck-at cells from the capacity point of view. For fixed integers $q > 1$ and $0 < s < q$, we consider the storage channel in which a cell can be partially stuck-at level s with probability p. This channel model is an example of a discrete memoryless memory cell which was studied in [7]. In the partially stuck-at cell model, we assume that a cell is partially stuck-at level s with some probability p. If a letter $x \in X = \{0, 1, \ldots, q-1\}$ is stored in a cell, then the letter $y \in Y = \{0, 1, \ldots, q-1\}$ is retrieved where $y = \max\{x, s\}$ with probability p.

If *both*, the encoder and decoder, know the state of the memory, i.e. the locations and levels of the partially stuck-at cells, then the capacity of this channel is

$$C_q(p, s) = 1-p+p\log_q(q-s) = 1-p\log_q\left(\frac{q}{q-s}\right). \quad (3)$$

To see this, notice that the lower bound on the redundancy from Theorem 2 holds also in this case as this is the number of all different memory states that can be programmed. The capacity is achievable since both the encoder and decoder know the locations of the partially stuck-at memory cells and thus can use all programmable memory states. According to [7], the capacity of any discrete memoryless memory cell channel in the case where *only the encoder* knows the memory state is the same as the one where both the encoder and decoder know the memory state. Thus, (3) is the capacity of the model studied in this work.

In the following, we want to analyze how close our constructions are to the capacity. Let us first consider the case $s = 1$. The construction based on binary codes from Section V can mask $u = pn$ partially stuck-at-1 memory cells if there exists an $[n, k, d]_2$ binary code that can mask $\widetilde{u} = \lfloor 2u/q \rfloor = \lfloor 2pn/q \rfloor$ binary usual stuck-at cells (which is the case if $d \geq \widetilde{u}+1$, see Theorem 1). Assume there exists a family of capacity-achieving linear binary codes with these properties, then we can use this family of codes in order to construct a family of PSMCs. Asymptotically, for large n and any q, the maximum achievable code rate of the construction from Theorem 6 approaches the value

$$R_q(p, 1) = \frac{n-r}{n} = 1 - \frac{2p}{q}\log_q\left(\frac{q}{\lfloor q/2 \rfloor}\right).$$

Similarly, for arbitrary s (for simplicity assume that $Q = s+1$ is a power of a prime), we obtain a family of PSMCs whose maximum achievable rate approaches the value

$$R_q(p, s) = 1 - \frac{2sp}{q}\log_q\left(\frac{q}{\lfloor q/(s+1) \rfloor}\right).$$

Finally, we conclude that the difference between the capacity and the redundancy of this construction is given by

$$C_q(p, s) - R_q(p, s) = p \cdot \underbrace{\left(\frac{2s}{q}\log_q\left(\frac{q}{\lfloor q/(s+1) \rfloor}\right) - \log_q\left(\frac{q}{q-s}\right)\right)}_{\overset{\text{def}}{=}\Delta(q,s)}, \quad (4)$$

where we call $\Delta(q, s)$ the *difference coefficient*.

Figure 1 illustrates the difference coefficient for some values of q and s. We see that the difference coefficient tends to zero for q large enough. The "plateaus" in the plots occur due to the floor operation in (4). A similar analysis can be done for the $(u, \mathbf{s})$-PSMC construction from Theorem 8.

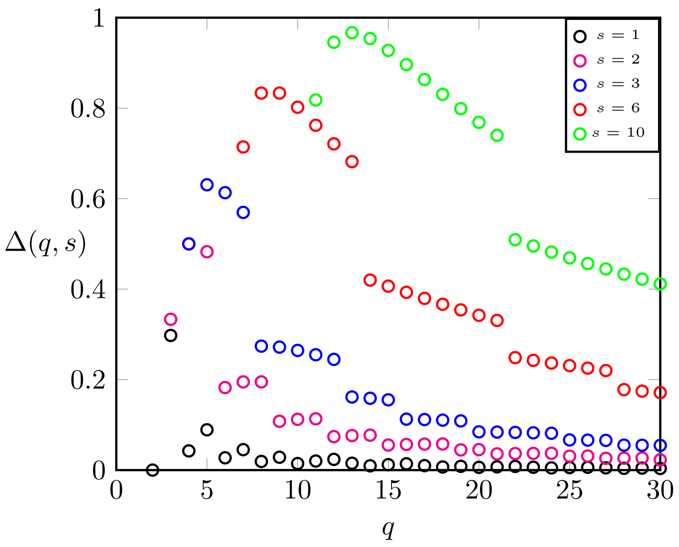

Figure 1. The difference coefficient $\Delta(q, s)$ for different values of q and s.

We note that the trivial construction from Theorem 2, in which only the levels $s, \ldots, q-1$ are used, can also be used to mask any number of cells which are partially stuck-at-s. The rate of this construction is $\log_q(q-s)$, and it is greater than the value of $R_q(p,s)$, when q is a multiple of $s+1$, if

$$\log_q(q-s) \geq 1 - \frac{2sp}{q}\log_q(s+1),$$

or

$$p \geq \frac{q}{2s}\log_{s+1}\left(\frac{q}{q-s}\right).$$

For example, for $s = 1$ we get that this threshold equals $\frac{q}{2}\log\left(\frac{q}{q-1}\right)$ and for q large enough, this value approaches $1/(2\ln 2) \approx 0.7213$.

We conclude that the maximum achievable rates, denoted by $R_q^{\max}(p,s)$, according to the constructions from Theorems 2, 6 and 8, when q is a multiple of $s+1$, is given by

$$R_q^{\max}(p,s) = \begin{cases} 1 - \frac{2sp}{q}\log_q(s+1) & \text{if } p \leq \frac{q}{2s}\log_{s+1}\left(\frac{q}{q-s}\right) \\ \log_q(q-s) & \text{else} \end{cases}$$

In Fig. 2, we plot the capacity $C(p,s)$ and the maximum achievable rates $R_q^{\max}(p,s)$ for $q = 8$ and $s = 1, 3$.

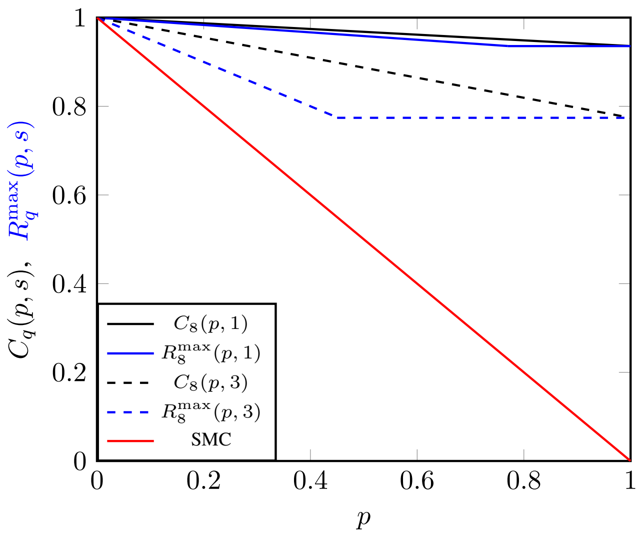

Figure 2. Capacity $C_q(p,s)$ & max. achievable rate $R_q^{\max}(p,s)$ for $q = 8$.

Notice that asymptotically, an SMC has rate $R = 1 - p$ and is therefore worse than $R_q^{\max}(p,s)$.

IX. Conclusion

In this paper, we have considered codes for masking partially stuck-at memory cells. After defining the defect model, we have derived lower and upper bounds on the minimum redundancy which is required to mask partially stuck-at cells in multilevel memories. We have presented two constructions of *partially stuck-at masking codes* (PSMCs). The first one masks any $u < q$ partially stuck-at cells, where q is the number of cell levels, and requires less than one redundancy symbol. When the cells are partially stuck-at level 1 (i.e., $s = 1$), this construction is asymptotically optimal. The second construction is based on using a binary error-correcting code and can be applied for any u and q. The constructions were first derived for $s = 1$ and then generalized to arbitrary stuck levels. We have further shown how these PSMCs can be applied for cells with unreachable levels (which can be seen as the dual problem to partially stuck-at cells) and that they outperform known code constructions for this model. Further, we have considered the capacity of the partially stuck-at channel and have analyzed the gap between the capacity and the achieved code rate. For $s = 1$, we achieve the capacity asymptotically.

Acknowledgement

A. Wachter-Zeh's work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 655109.

E. Yaakobi's work has been supported in part by the Israel Science Foundation (ISF) grant No. 1624/14.

References

[1] I. Belov and A. Shashin, "Codes that correct triple defects in memory (in Russian)," *Prob. Inf. Transm.*, vol. 13, no. 4, pp. 62–65, 1977.
[2] J. Borden and A. Vinck, "On coding for stuck-at defects," *IEEE Trans. Inform. Theory*, vol. 33, no. 5, pp. 729–735, Sep. 1987.
[3] C. L. Chen, "Linear Codes for masking memory defects," *IEEE Trans. Inform. Theory*, vol. 31, no. 1, pp. 105–106, Jan. 1985.
[4] J. Fridrich, M. Goljan, and D. Soukal, "Steganography via codes for memory with defective cells," in *Allerton Conference on Communication, Control and Computing*, Sep. 2005.
[5] R. Gabrys, F. Sala, and L. Dolecek, "Coding for unreliable flash memory cells," *IEEE Comm. Letters*, vol. 18, no. 9, pp. 1491–1494, Jul. 2014.
[6] C. Heegard, "Partitioned linear block codes for computer memory with stuck-at defects," *IEEE Trans. Inform. Theory*, vol. 29, no. 6, pp. 831–842, Nov. 1983.
[7] C. Heegard and A. A. El Gamal, "On the capacity of computer memory with defects," *IEEE Trans. Inform. Theory*, vol. 29, no. 5, pp. 731–739, Sep. 1983.
[8] Y. Kim and B. Kumar, "Coding for memory with stuck-at defects," in *IEEE Int. Conf. Communications*, Jun. 2013, pp. 4347–4352.
[9] A. V. Kuznetsov, T. Kasami, and S. Yamamura, "An error correcting scheme for defective memory," *IEEE Trans. Inform. Theory*, vol. 24, no. 6, pp. 712–718, Nov. 1978.
[10] A. Kuznetsov, "Coding in a channel with generalized defects and random errors (in Russian)," *Probl. Inf. Transm.*, vol. 21, no. 1, pp. 28–34, 1985.
[11] A. Kuznetsov and B. Tsybakov, "Coding for memories with defective cells (in Russian)," *Probl. Inf. Transm.*, vol. 10, no. 2, pp. 52–60, 1974.
[12] L. A. Lastras-Montaño, A. Jagmohan, and M. M. Franceschini, "Algorithms for memories with stuck cells," in *IEEE Int. Symb. Inf. Theory*, Jun. 2010, pp. 968–972.
[13] L. Lastras-Montaño, A. Jagmohan, and M. Franceschini, "An area and latency assessment for coding for memories with stuck cells," in *Workshop Appl. Comm. Theo. to Em. Memory Techn.*, Dec. 2010.
[14] V. Losev, V. Konopel'ko, and Y. Daryakin, "Double-and-triple-defect-correcting codes (in Russian)," *Prob. Inf. Transm.*, vol. 14, no. 4, pp. 98–101, 1978.
[15] N. Seong, D. Woo, V. Srinivasan, J. Rivers, and H. Lee, "SAFER: stuck-at-fault error recovery for memories," in *MICRO-43*, Dec. 2010.
[16] B. S. Tsybakov, S. I. Gelfand, A. V. Kuznetsov, and S. I. Ortyukov, "Reliable computation and reliable storage of information," in *IEEE-USSR Workshop*, Dec. 1975.
[17] B. Tsybakov, "Defects and error correction (in Russian)," *Prob. Inf. Transm.*, vol. 11, no. 1, pp. 21–30, 1975.
[18] ——, "Group additive defect-correcting codes (in Russian)," *Prob. Inf. Transm.*, vol. 11, no. 1, pp. 111–113, 1975.
[19] A. Wachter-Zeh and E. Yaakobi, "Codes for Partially Stuck-at Memory Cells," in *Int. ITG Conf. on Systems, Communications and Coding (SCC)*, Feb. 2015.
[20] ——, "Codes for Partially Stuck-at Memory Cells," *preprint*, May 2015. [Online]. Available: http://arxiv.org/abs/1505.03281