Codes for Partially Stuck-At Memory Cells

Antonia Wachter-Zeh, *Member, IEEE*, and Eitan Yaakobi, *Member, IEEE*

***Abstract*— In this paper, we study a new model of defect memory cells, called partially stuck-at memory cells, which is motivated by the behavior of multi-level cells in non-volatile memories, such as flash memories and phase change 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. Our main contribution in this paper is the study of codes for masking u partially stuck-at cells. We first derive lower and upper bounds on the redundancy of such codes. The upper bounds are based on two trivial constructions. We then present three code constructions over an alphabet of size q, by first considering the case where the cells are partially stuck-at level $s = 1$. The first construction works for $u < q$ and is asymptotically optimal if $u+1$ divides q. The second construction uses the reduced row echelon form of matrices to generate codes for the case $u \geq q$, and the third construction solves the case of arbitrary u by using codes, which mask binary stuck-at cells. We then show how to generalize all constructions to arbitrary stuck levels. Furthermore, we study 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. Last, 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, defect cells, flash memories, phase change memories.**

I. Introduction

NON-VOLATILE memories such as flash memories and phase change memories (PCMs) have paved their way to be a dominant memory solution for a wide range of applications, from consumer electronics to solid state drives. The rapid increase in the capacity of these memories along with the introduction of multi-level technologies have significantly reduced their cost. However, at the same time, their radically degraded reliability has demanded for advanced signal processing and coding solutions.

The cells of PCMs can take distinct physical states. In the simplest case, a PCM cell has two possible states, an amorphous state and a crystalline state. Multi-level PCM cells can be designed by using partially crystalline states [3]. Failures of PCM cells stem from the heating and cooling processes of the cells. These processes may prevent a PCM cell to switch between its states and thus the cells become *stuck-at* [9], [13], [20], [23]. Similarly, in the multi-level setup, a cell can get stuck-at one of the two extreme states, amorphous or crystalline. Or, alternatively, the cells cannot be programmed to a certain state, but can be in all other ones; for example, if a cell cannot be programmed to the amorphous state it can still be at the crystalline state as well as all intermediate ones. A similar phenomenon appears in flash memories. Here, the information 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. [17]), the memory consists of n binary cells of which u are stuck-at either in the zero or one state. A cell is said to be *stuck-at level* $s \in \{0, 1\}$ if the value of the cell cannot be changed. Therefore, only data can be written into the memory which matches the fixed values at the stuck-at cells. A *code for stuck-at cells* maps message vectors on 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. Hence, 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. If u cells are stuck-at, then it is not possible to encode more than $n-u$ bits, and thus the problem is to design codes with redundancy close to u. The same problem is relevant for the non-binary setup of this model, where the cells can be stuck at any level.

The study of codes for memories with stuck-at cells, also known as *memories with defects*, takes a while back to the 1970s. To the best of our knowledge, the problem was first studied in 1974 by Kuznetsov and Tsybakov [17]. Since then, several more papers have appeared, e.g. [1], [2], [6], [11], [15], [16], [21], [28]–[30]. 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 PCM 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. [7], [14], [18], [19], [25].

Manuscript received May 13, 2015; revised September 16, 2015; accepted December 9, 2015. Date of publication December 31, 2015; date of current version January 18, 2016. A. Wachter-Zeh was supported by the European Union's Horizon 2020 Research and Innovation Programme under the Marie Sklodowska-Curie under Grant 655109. E. Yaakobi was supported by the Israel Science Foundation under Grant 1624/14. This paper was presented at the 2015 10th International ITG Conference on Systems, Communications and Coding and the 53rd Annual Allerton Conference on Communication, Control, and Computing.

The authors are with the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: antonia@cs.technion.ac.il; yaakobi@cs.technion.ac.il).

Communicated by M. Schwartz, Associate Editor for Coding Techniques.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2015.2512581

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 greater than or equal to s. In this work, we provide upper and lower bounds on the redundancy and give several code constructions with small redundancy in order to mask partially stuck-at cells. To facilitate the explanations, our constructions are first given for the case $s = 1$, and later generalized to arbitrary levels.

In the first part of the paper, we derive lower and upper bounds on the redundancy of codes that mask partially stuck-at memory cells. The first lower bound is based on the number of states that each partially stuck-at cell can attain. For the case where all cells are partially stuck-at the same level s, we also derive an improved lower bound. Further, codes which mask any number of partially stuck-at cells can simply be constructed by using only the levels $\max_i s_i, \ldots, q-1$ in each cell. Alternatively, codes which mask u stuck-at cells (not partially) can be used to mask u partially stuck-at cells. Thus, these two schemes provide an upper bound on the minimum redundancy of codes that is necessary to mask u partially stuck-at cells.

In the course of this paper, we provide several code constructions and analyze how far they are from our lower and upper bounds on the redundancy. In particular, for $u < q$, we first show that one redundancy symbol is sufficient to mask all partially stuck-at cells. Further, an improvement of this construction turns out to be asymptotically optimal in terms of the redundancy if $u+1$ divides q. Our second construction uses the parity-check matrix of q-ary error-correcting codes. The third 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 rest of the paper is organized as follows. In Section II, we introduce 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 give a construction based on q-ary codes and in Section VI we show a more general solution by using codes which mask binary stuck-at cells. Section VII 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 VIII and Section IX analyzes the capacity of the partially stuck-at channel. Finally, Section X concludes this paper.

II. Definitions and Preliminaries

A. Notations

In this section, we formally define the models of (partially) stuck-at cells studied in this paper and introduce the notations and tools we will use in the sequel. In general, 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 assumed to represent q-ary symbols having values belonging to the set $[q]$. We can therefore represent the memory cells 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-***s**, where $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1})$ is a vector and contains the different stuck levels $s_0, s_1, \ldots, s_{u-1}$ 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 $[q]$ (in which case q is a power of a prime and $[q]$ corresponds to the finite field $\mathbb{F}_q$ of order 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 $\{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n]$, the stuck levels $s_0, s_1, \ldots, s_{u-1} \in [q]$ 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.,*

$$y^{(m)}_{\phi_i} = s_i, \quad \forall i \in [0, u-1],$$

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 $\{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n]$, the partially stuck levels $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$ 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.,

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

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

The term *masking* refers to the process of finding a codeword that matches the levels of the (partially) stuck cells and can therefore be written correctly to the memory.

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. Furthermore, in the design of codes which mask stuck-at cells, the encoder and decoder need to know in advance only the number of stuck-at cells u. However, for codes that mask partially stuck-at cells, the encoder and decoder need to know both the number of partially stuck-at cells u as well as their partially stuck-at levels, which are specified by the vector stored in $\mathbf{s}$. Hence, Construction 1 of an SMC (presented later) does not depend on the explicit stuck levels (only on the number of stuck cells), whereas our constructions and their redundancies depend on the explicit partially stuck-at levels (stored in the vector $\mathbf{s}$).

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 simply we speak about a u-PSMC, we mean a code which masks at most u partially-stuck-at-1 cells.

In the design of (P)SMCs, the goal is therefore 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 $[q]$. Note that throughout this paper we only consider the problem of masking partially stuck-at cells, without correcting additional (random) errors as e.g. in [11] for stuck-at cells.

C. Codes for Stuck-At Cells

Codes for masking stuck-at cells were studied before and one such construction and its encoding and decoding algorithms are shown in the following; see e.g. [11]. We show the proof of the properties of the construction for the completeness of results in this paper and since our encoding and decoding algorithms use similar ideas.

Construction 1 (Masking Stuck-At Cells, [11]): Let $\mathbf{H}$ be a systematic $(n-k) \times n$ parity-check matrix of a code $\mathcal{C}$ which is known to both, encoder and decoder. We define a u-SMC by Algorithms 1 and 2.

Theorem 1: Let $\mathcal{C}$ be an $[n, k, d]_q$ code with minimum distance $d \geq u+1$, where q is a prime power. Then, Construction 1 is a u-SMC with redundancy $r = n - k$.

Proof: We assume that $\mathbf{H}$ is a systematic $(n-k) \times n$ parity-check matrix of the code $\mathcal{C}$ which is known to both, encoder and decoder.

Algorithm 1 describes the encoding process. We have to prove that in Step 2, a vector $\mathbf{z}$ always exists such that the encoded vector $\mathbf{y}$ masks the u stuck-at cells. Define $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) = \mathbf{z} \cdot \mathbf{H}$. We have to fulfill the following u equations:

$$w_{\phi_i} + x_{\phi_i} = s_i, \quad \forall i \in [u].$$

Denote by $\mathbf{H}_u$ the u columns of $\mathbf{H}$ indexed by $\phi_0, \ldots, \phi_{u-1}$. Then, we have to find $\mathbf{z}$ such that

$$\mathbf{z} \cdot \mathbf{H}_u = (s_0 - w_{\phi_0}, s_1 - w_{\phi_1}, \ldots, s_{u-1} - w_{\phi_{u-1}}).$$

This is a heterogeneous linear system of equations with at most $u \leq d-1 \leq n-k$ linearly independent equations and $n-k$ unknowns (the elements of $\mathbf{z}$). Therefore, there is at least one solution for the vector $\mathbf{z}$ such that $\mathbf{y}$ masks the stuck-at cells.

Algorithm 2 describes the decoding process. We have to prove that $\widehat{\mathbf{m}} = \mathbf{m}$. Note that $\mathbf{H}$ is a *systematic* parity-check matrix and therefore, the first $n-k$ positions of the output vector $\mathbf{y}$ of Algorithm 1 (which is at the same time the input of Algorithm 2) equal $\mathbf{z}$. Therefore, in Step 1 in Algorithm 2, we obtain $\widehat{\mathbf{z}} = \mathbf{z}$ and thus, in Step 2, $\widehat{\mathbf{w}} = \mathbf{y} - \mathbf{z} \cdot \mathbf{H} = \mathbf{w}$ and in Step 3, $\widehat{\mathbf{m}} = \mathbf{m}$. ■

It is not completely known whether Construction 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.

Let us illustrate Construction 1 with an example.

Example 1: Let $u = 2$, $q = 3$ and $n = 5$. Therefore, we need a ternary code of minimum distance at least $u + 1 = 3$. The largest such code is a $[5, 2, 3]_3$ code (see [10]) with a

Algorithm 1 ENCODING-1$(\mathbf{m}; \phi_0, \phi_1, \ldots, \phi_{u-1};\ s_0, s_1, \ldots, s_{u-1})$

Input: • message: $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1}) \in [q]^k$
• positions of stuck-at cells: $\{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n]$
• levels of stuck-at cells: $s_0, s_1, \ldots, s_{u-1} \in [q]$

1 $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1}) \leftarrow (\mathbf{0}_{n-k}, m_0, m_1, \ldots m_{k-1})$
2 Find $\mathbf{z} \in [q]^{n-k}$ such that $\mathbf{y} \leftarrow \mathbf{w} + \mathbf{z} \cdot \mathbf{H}$ masks the stuck-at cells[1]

Output: vector $\mathbf{y}$ with $y_{\phi_i} = s_i$, $\forall i \in [u]$

Algorithm 2 DECODING-1$(\mathbf{y})$

Input: • stored vector: $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in [q]^n$

1 $\widehat{\mathbf{z}} \leftarrow (y_0, y_1, \ldots, y_{n-k-1})$
2 $\widehat{\mathbf{w}} = (\widehat{w}_0, \widehat{w}_1, \ldots, \widehat{w}_{n-1}) \leftarrow \mathbf{y} - \widehat{\mathbf{z}} \cdot \mathbf{H}$
3 $\widehat{\mathbf{m}} \leftarrow (\widehat{w}_{n-k}, \widehat{w}_{n-k+1}, \ldots, \widehat{w}_{n-1})$

Output: message vector $\widehat{\mathbf{m}} \in [q]^k$

[1]We assume here and in the rest of the paper that there is a one-to-one mapping $F : [q] \to \mathbb{F}_q$ so arithmetic operations like multiplication and addition in $\mathbf{w} + \mathbf{z} \cdot \mathbf{H}$ are defined as $F(\mathbf{w}) + F(\mathbf{z}) \cdot \mathbf{H}$.

systematic parity-check matrix

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

We will show how to construct a $(5, 3^2 = 9)_3$ *2-SMC which masks any* $u' \leq u = 2$ *stuck-at cells.*

For the encoding, we proceed as in Algorithm 1. Let $\mathbf{m} = (m_0, m_1) = (2, 1)$ *be the information we want to store and assume that the cells at positions* $\phi_0 = 0$ *and* $\phi_1 = 4$ *are stuck-at* $s_0 = 1$ *and* $s_1 = 2$*. Further, let* $\mathbf{w} = (0, 0, 0, m_0, m_1) = (0, 0, 0, 2, 1)$*. Given* $\mathbf{w}$*,* ϕ_0*,* ϕ_1 *and* s_0*,* s_1*, our goal is to find a vector* $\mathbf{x} \in [3]^5$ *such that* $\mathbf{y} \equiv (\mathbf{w} + \mathbf{x}) \bmod 3$ *matches the partially stuck-at cells and yet the information stored in* $\mathbf{m}$ *can be reconstructed from* $\mathbf{y}$*.*

We use $\mathbf{z} = (1, 0, 1)$*. Then,* $\mathbf{x} = \mathbf{z} \cdot \mathbf{H} = (1, 0, 1, 1, 1)$ *and* $\mathbf{y} = \mathbf{w} + \mathbf{x} = (1, 0, 1, 0, 2)$ *which masks the stuck-at cells.*

To reconstruct $\mathbf{w}$*, given* $\mathbf{y}$*, we notice that* $(y_0, y_1, y_2) = \mathbf{z} = (1, 0, 1)$ *and we can simply calculate* $\mathbf{y} - (y_0, y_1, y_2) \cdot \mathbf{H}$ *to obtain* $\mathbf{w}$ *and therefore* $\mathbf{m}$*.*

Thus, the required redundancy for masking $u = 2$ *stuck-at cells with Theorem 1 is* $r = 3$ *(the first three symbols).*

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 $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$. Hence, when a cell at position ϕ_i is partially stuck-at level s_i, it can only store the values $\{s_i, s_i + 1, \ldots, q-1\}$.

Theorem 2 (Bounds on the Redundancy): For any number of $u \leq n$ *partially stuck-at cells and any levels* $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$*, the value of the minimum redundancy* $r_q(n, u, \mathbf{s})$ *to mask these cells satisfies*

$$\begin{aligned} & u - \log_q \left(\prod_{i=0}^{u-1} (q - s_i) \right) \\ & \quad \leq r_q(n, u, \mathbf{s}) \\ & \quad \leq \min \left\{ n \cdot \left(1 - \log_q \left(q - \max_i \{s_i\}\right)\right),\ \rho_q(n, u+1) \right\}. \end{aligned}$$

Proof: Let us start with 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 $\max_i\{s_i\}, \ldots, q-1$ 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

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

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

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 will serve as a reference for our constructions, as we should ensure to construct codes with smaller redundancy than the upper bound and study how far their redundancy is from the lower bound.

For the special case of partially stuck-at-1 cells, we obtain the following bounds on the redundancy $r_q(n, u, \mathbf{1})$ of a u-PSMC for all positive integers n, q, u where $u \leq n$:

$$\begin{aligned} & u\left(1 - \log_q(q-1)\right) \\ & \quad \leq r_q(n, u, \mathbf{1}) \\ & \quad \leq \min\left\{ n\left(1 - \log_q(q-1)\right),\ \rho_q(n, u+1) \right\}. \end{aligned} \tag{1}$$

For partially stuck-at-s cells, we further improve the lower bound on the redundancy in the next theorem.

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

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

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

$$\begin{aligned} r_q(n, u, s) & \geq n - \log_q \left(\frac{q^n + u(q-s)^n}{u+1} \right) \\ & = \log_q(u+1) - \log_q\left(1 + u(1 - s/q)^n\right). \end{aligned}$$

∎

For $u = 1$, we obtain from Theorem 3:

$$r_q(n, 1, s) \geq \log_q 2 - \log_q\left(1 + (1 - s/q)^n\right).$$

For $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1})$ and $s_{max} = \max_{i\in[u]}\{s_i\}$ and $s_{min} = \min_{i\in[u]}\{s_i\}$, we have

$$r_q(n, u, s_{max}) \geq r_q(n, u, \mathbf{s}) \geq r_q(n, u, s_{min}), \tag{2}$$

so we can always use $r_q(n, u, 1)$ as a lower bound for partially stuck-at-$s_0, s_1, \ldots, s_{u-1}$ cells.

Examples 2 and 3 show cases where the lower bound from Theorem 3 is better than the lower bound from (1).

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

A. Code Construction

In this section, we show a simple construction of u-PSMCs for masking any $u < q$ partially stuck-at-1 cells, which works for any q (not necessarily a prime power). Let us illustrate the idea of this construction with an example.

Example 2: Let $u = 2$, $q = 3$ and $n = 5$. We will show how to construct a $(5, 3^4 = 81)_3$ u-PSMC.

Let $\mathbf{m} = (m_0, m_1, m_2, m_3) = (2, 0, 1, 0)$ be the information we want to store and assume that the two cells at positions $\phi_0 = 1$ and $\phi_1 = 2$ are partially stuck-at-1. We set $\mathbf{w} = (0, m_0, m_1, m_2, m_3) = (0, 2, 0, 1, 0)$. Given $\mathbf{w}$, ϕ_0 and ϕ_1, our goal is to find a vector $\mathbf{x}$ such that $\mathbf{y} \equiv (\mathbf{w} + \mathbf{x}) \bmod 3$ masks the partially stuck-at-1 cells and yet the information stored in $\mathbf{m}$ can be reconstructed from $\mathbf{y}$.

We use $\mathbf{x} = z \cdot \mathbf{1}_5$, for some $z \in [3]$. Since $w_1 = 2$, we can choose any value from $[3]$ for z except for 1, and since $w_2 = 0$, we can choose any value but 0. Thus, we choose $z = 2$ and encode the vector $\mathbf{y}$ to be $\mathbf{y} = \mathbf{w} + 2 \cdot \mathbf{1}_5 = (2, 1, 2, 0, 2)$, which masks the two partially stuck-at-1 cells.

To reconstruct $\mathbf{w}$, given $\mathbf{y}$, notice that $y_0 = z = 2$ and we can simply calculate $\mathbf{y} - y_0 \cdot \mathbf{1}_5$ to obtain $\mathbf{w}$ and therefore $\mathbf{m}$.

Thus, the required redundancy for masking $u = 2$ partially stuck-at-1 cells is $r = 1$ (the first symbol). The lower bound from (1) is $2 \cdot (1 - \log_3 2) = 0.738$, the lower bound from Theorem 3 gives 0.787 (and therefore improves upon the bound from Theorem 2) and the upper bound is $\min\{5 \cdot (1 - \log_3 2),\ \rho_3(5, 3) = 3\} = 1.845$, where the second part was shown in Example 1.

Construction 2: Let $u < q$ and $u \leq n$. We define a u-PSMC over $[q]$ by Algorithms 3 and 4.

The following theorem shows that this construction yields u-PSMCs with small redundancy for any $u < q$.

Theorem 4: If $u < q$ and $u \leq n$, then for all n, Construction 2 provides an $(n, M = q^{n-1})_q$ u-PSMC with redundancy of one symbol.

Proof: The encoding is shown in Algorithm 3. Let $\phi_0, \phi_1, \ldots, \phi_{u-1}$ be the positions of some u partially stuck-at-1 cells and let $\mathbf{m} = (m_0, m_1, \ldots, m_{n-2}) \in [q]^{n-1}$ be the message to be written to the memory $\mathbf{w}$. We first set $\mathbf{w} = (0, m_0, m_1, \ldots, m_{n-2})$. Since $u < q$, there exists at least one value $v \in [q]$ such that $w_{\phi_0}, w_{\phi_1}, \ldots, w_{\phi_{u-1}} \neq v$. Choose $z = q - v \equiv -v \bmod q$ and therefore, $w_{\phi_i} + z \equiv (w_{\phi_i} - v) \bmod q \neq 0$. Thus, the output vector $\mathbf{y}$ masks all u partially stuck-at-1 cells.

The decoding algorithm is shown in Algorithm 4 and it simply uses the fact that y_0 is used to store the value of z. Thus, $\widehat{\mathbf{m}} = \mathbf{m}$.

Algorithm 3 ENCODING-2$(\mathbf{m}; \phi_0, \phi_1, \ldots, \phi_{u-1})$

Input: • message: $\mathbf{m} = (m_0, m_1, \ldots, m_{n-2}) \in [q]^{n-1}$
• positions of partially 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 $v \in [q]$ such that $v \notin \{w_{\phi_0}, w_{\phi_1}, \ldots, w_{\phi_{u-1}}\}$
3 $z \leftarrow q - v$
4 $\mathbf{y} \leftarrow (\mathbf{w} + z \cdot \mathbf{1}_n) \bmod q$
Output: vector $\mathbf{y}$ with $y_{\phi_i} \geq 1$, $\forall i \in [u]$

Algorithm 4 DECODING-2$(\mathbf{y})$

Input: • stored vector: $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in [q]^n$

1 $\widehat{z} \leftarrow y_0$
2 $\widehat{\mathbf{m}} \leftarrow ((y_1, y_2, \ldots, y_{n-1}) - \widehat{z} \cdot \mathbf{1}_n) \bmod q$
Output: message vector $\widehat{\mathbf{m}} \in [q]^{n-1}$

Lastly, since we need only one symbol to store the value of $z = q - v$ in the first cell, the redundancy is one. ■

This principle works for *any* n since $\mathbf{1}_n$ always exists. Further, q does not have to be a prime power and we can use this strategy even if $[q]$ does not constitute a finite field.

Construction 2 provides a significant improvement compared to codes for the (stronger) model of usual stuck-at cells, where the required redundancy is $\rho_q(n, u + 1) \geq u$ (compare Construction 1). In order to compare our result to the first term in the upper bound from (1), we use the approximation $\ln(q) - \ln(q - 1) \approx 1/q$ for q large enough and thus,

$$1 - \log_q(q - 1) \approx \frac{1}{q \ln q}.$$

Hence, for $n \geq q \ln q$, Construction 2 achieves smaller redundancy than the upper bound from (1). However, for $u < q$ and n large enough the lower bound on the redundancy from Theorem 3 is $\log_q(u + 1) < 1$. This suggests that codes with better redundancy might exist, as we shall see in the next subsection.

B. Improvement of Construction 2

In this subsection, we show how to improve Construction 2 and therefore, how to reduce the required redundancy. Let us first show an example of this modification.

Example 3: Let $u = 2$ and $q = 6$. Then, with the same principle as in Algorithm 1 and $z \in [3]$, we can mask any two partially stuck-at-1 cells with one redundancy symbol. The important observation here is that z can always be chosen from a small subset of $[q]$. Assume that the first cell stores the redundancy, namely $y_0 = z$, and takes therefore only three out of six possible values. We can store additional information in this cell e.g. as follows: 0 is represented by 0 or 3; 1 is represented by 1 or 4; and 2 is represented by 2 or 5. By choosing e.g. between storing 1 or 4, we store additional information in the redundancy cell. Thus, the information

Algorithm 5 ENCODING-3$\big(\mathbf{m}; m'; \phi_0, \phi_1, \ldots, \phi_{u-1}\big)$

Input: • message: $\mathbf{m} = (m_0, m_1, \ldots, m_{n-2}) \in [q]^{n-1}$
• additional message: $m' \in \left[\left\lfloor \frac{q}{u+1} \right\rfloor\right]$
• 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 $v \notin \big\{w_{\phi_0} \bmod (u+1), \ldots, w_{\phi_{u-1}} \bmod (u+1)\big\}$
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 $y_{\phi_i} \geq 1$, $\forall i \in [u]$

Algorithm 6 DECODING-3 $\big(\mathbf{y}\big)$

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 $\widehat{m'} \leftarrow \dfrac{q - \widehat{z} - \widehat{v}}{u+1}$
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 $\widehat{m'}$

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* (1) *gives* ≈ 0.204. *The lower bound from Theorem 3 gives* 0.284 *for* $n = 5$, *and* 0.457 *for* $n = 10$. *For* $n \to \infty$, *the lower bound from Theorem 3 reaches* 0.613 *which is exactly the redundancy achieved by our construction.*

In general, assume that $u < q$, then there is some $v \in [u+1]$ such that $w_{\phi_0}, w_{\phi_1}, \ldots, w_{\phi_{u-1}} \neq v$. Hence, if u is significantly smaller than q, then there is more than one value to choose the value of v from and the redundancy decreases by $\log_q(\lfloor q/(u+1) \rfloor)$ symbols.

With these considerations, we obtain the following result.

Construction 3: Let $u < q$ and $u \leq n$. We define a u-PSMC over $[q]$ by Algorithms 5 and 6.

Theorem 5: If $u < q$ and $u \leq n$, then for all n, Construction 3 provides 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 modify Algorithm 3 to obtain the encoding procedure for Construction 3.

The encoding algorithm is a generalization of Algorithm 3. We will prove that the output vector $\mathbf{y}$ masks the partially stuck-at-1 cells. In Step 2, we can always find a value $v \in [u+1]$ as required since the set $\big\{w_{\phi_0} \bmod (u+1), \ldots, w_{\phi_{u-1}} \bmod (u+1)\big\}$ 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, we conclude that $y_{\phi_i} \neq 0$, for all $i \in [u]$, and the partially stuck-at-1 positions are masked.

The decoding algorithm has to guarantee that we can recover $\mathbf{m}$ and m' and is shown in Algorithm 6. 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. ∎

Clearly, for $u < q$, the redundancy is $r \leq 1$. The following lemma shows that this construction is asymptotically optimal.

Theorem 6 (Optimality of Construction 3): If $(u+1)$ divides q, the u-PSMC from Construction 3 is asymptotically optimal in terms of the redundancy.

Proof: If $(u+1)|q$, then the redundancy from Theorem 5 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. CONSTRUCTIONS USING q-ARY CODES FOR PARTIALLY STUCK-AT LEVEL $s = 1$ CELLS

For $u \geq q$, it is not always possible to construct u-PSMCs with Construction 3. Therefore, this section provides a construction which works well for $u \geq q$ and can be seen as a generalization of Construction 2 (instead of the all-one vector, we use a parity-check matrix). Let us start with a general statement for any u and any prime power q.

Theorem 7: Let q be a prime power and let a $\kappa \times n$ matrix $\mathbf{H} = \big(H_{i,j}\big)_{j\in[n]}^{i\in[\kappa]}$ over $\mathbb{F}_q$ be given. If the reduced row Echelon (RRE) form of any $\kappa \times u$ submatrix (denoted by $\mathbf{H}^{(u)}$) has the following form (up to column permutations)[2]:

$$\mathrm{RRE}(\mathbf{H}^{(u)}) = \begin{pmatrix} \underbrace{1\ \bullet\ \ldots\ \bullet}_{\leq q-1} & \circ\ \circ\ \ldots\ \circ & \ldots & \ldots & \ldots\ \circ \\ & \underbrace{1\ \bullet\ \ldots\ \bullet}_{\leq q-1} & \circ\ \circ\ \ldots\ \circ & \ldots & \ldots\ \circ \\ & & \underbrace{1\ \bullet\ \ldots\ \bullet}_{\leq q-1} & \circ\ \ldots & \ldots\ \circ \\ & & & \ddots & \vdots \\ & \mathbf{0} & & & \underbrace{1\ \bullet\ \ldots\ \bullet}_{\leq q-1} \end{pmatrix}, \tag{3}$$

where $\bullet$ has to be a non-zero element from $\mathbb{F}_q$ and $\circ$ is any element from $\mathbb{F}_q$, then, there exists a u-PSMC over $\mathbb{F}_q$ of length n and redundancy $r = \kappa$.

[2]Notice that $\mathrm{RRE}(\mathbf{H}^{(u)})$ can have less than κ rows.

Proof: Assume w.l.o.g. that the partially stuck-at-1 positions are $[u]$. As before, if q is a prime power, we fix a mapping from the elements of the extension field $\mathbb{F}_q$ to the set of integers $[q]$. The encoding and decoding follow similar steps as Algorithm 1. From a given message vector $\mathbf{m} \in [q]^{n-\kappa}$, we first define a vector $\mathbf{w} = (\mathbf{0}_\kappa, \mathbf{m})$. Then, we search a vector $\mathbf{z}$ such that $\mathbf{y} = \mathbf{w} + \mathbf{z} \cdot \mathbf{H}$ masks all the partially stuck-at-1 cells.

First, assume that RRE($\mathbf{H}$) can mask any u partially stuck-at-1 cells and let us show that then also $\mathbf{H}$ can mask any u partially stuck-at-1 cells. By assumption, there is a vector $\mathbf{z} = (z_0, z_1, \ldots, z_{\kappa-1})$ such that $\mathbf{y} = \mathbf{w} + \mathbf{z} \cdot \mathrm{RRE}(\mathbf{H})$ masks all the partially stuck-at-1 cells. Since $\mathbf{H} = \mathbf{T} \cdot \mathrm{RRE}(\mathbf{H})$, for some $\kappa \times \kappa$ full-rank matrix $\mathbf{T}$, the vector $\widetilde{\mathbf{z}} = \mathbf{z} \cdot \mathbf{T}^{-1}$ masks the same stuck-at cells when multiplied by $\mathbf{H}$ since $\widetilde{\mathbf{z}} \cdot \mathbf{H} = \mathbf{z} \cdot \mathbf{T}^{-1}\mathbf{H} = \mathbf{z} \cdot \mathrm{RRE}(\mathbf{H})$.

Second, we prove that RRE($\mathbf{H}$) can mask any u partially stuck-at-1 cells. To simplify notations, assume w.l.o.g. that each "block" of RRE($\mathbf{H}^{(u)}$) has length exactly $q-1$, if is shorter it is clear that the principle also works. Similar to the proof of Theorem 4, there is (at least) one value $z_0 \in \mathbb{F}_q$ such that

$$z_0 \cdot H_{0,i} \neq -w_i, \quad \forall i \in [q-1], \tag{4}$$

since (4) consists of (at most) $q-1$ constraints and there are q possible values for z_0. Hence, $(z_0, z_1, \ldots, z_{\kappa-1}) \cdot \mathrm{RRE}(\mathbf{H}^{(u)})$ will mask the first $q-1$ partially stuck-at-1 cells for any $z_1, \ldots, z_{\kappa-1}$ since $w_i + z_0 \cdot H_{0,i} \neq 0$, for all $i = 0, \ldots, q-2$.

Similarly, there is (at least) one value $z_1 \in \mathbb{F}_q$ such that

$$z_1 \cdot H_{1,i} \neq -(w_i + z_0 \cdot H_{0,i}), \quad \forall i \in [q-1, 2q-3],$$

and $(z_0, z_1, \ldots, z_{\kappa-1}) \cdot \mathrm{RRE}(\mathbf{H}^{(u)})$ will mask the second $q-1$ partially stuck-at-1 cells for any $z_2, \ldots, z_{\kappa-1}$. This principle can be continued for each block of $q-1$ cells and clearly, column permutations of RRE($\mathbf{H}^{(u)}$) pose no problem on this strategy. ∎

Even though Theorem 7 provides a general scheme to construct u-PSMCs, it is not necessarily clear how to choose the matrix $\mathbf{H}$ in a way that it satisfies both the properties specified in the theorem and provides codes with good redundancy. We will next show an example of a simple application of this theorem and then provide a construction of u-PSMCs with low redundancy. Note also that in contrast to Theorem 4, it is not clear if a similar statement as Theorem 7 also holds if q is not a prime power.

Example 4: Based on Theorem 7, the following $\kappa \times n$ matrix, where $\kappa = \lceil \frac{n}{q-1} \rceil$, can clearly mask up to $u = n$ partially stuck-at-1 cells over $\mathbb{F}_q$:

$$\begin{pmatrix} \underbrace{1\,1\,\ldots\,1}_{q-1} & & & & \\ & \underbrace{1\,1\,\ldots\,1}_{q-1} & & 0 & \\ & & \underbrace{1\,1\,\ldots\,1}_{q-1} & \ddots & \\ & 0 & & & \underbrace{1\,1\,\ldots\,1}_{\leq q-1} \end{pmatrix}.$$

The redundancy of this code for masking any $u \leq n$ stuck-at-one cells is $r = \kappa = \lceil \frac{n}{q-1} \rceil$. Thus, for $q \geq n+1$, this provides codes for masking up to $u = n$ partially stuck-at-1 cells with redundancy $r = 1$. However, for $u = n$ partially stuck-at-1 cells, we can always use the trivial construction from Section II, which outperforms the previous matrix for all $q > 2$.

Algorithm 7 ENCODING-4$(\mathbf{m}; \phi_0, \phi_1, \ldots, \phi_{u-1})$

Input: • message: $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1}) \in [q]^k$
• positions of partially stuck-at-1 cells: $\{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n]$

1 $\mathbf{w} = (w_0, w_1, \ldots, w_{n-1}) \leftarrow (\mathbf{0}_{n-k}, m_0, m_1, \ldots m_{k-1})$
2 Find $\mathbf{z} = (z_0, z_1, \ldots, z_{n-k-1})$ as explained in the proof of Theorem 7
3 $\mathbf{y} \leftarrow \mathbf{w} + \mathbf{z} \cdot \mathbf{H}$
Output: vector $\mathbf{y}$ with $y_{\phi_i} \geq 1$, $\forall i \in [u]$

Algorithm 8 DECODING-4$(\mathbf{y})$

Input: • stored vector: $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in [q]^n$

1 $\widehat{\mathbf{z}} \leftarrow (y_0, y_1, \ldots, y_{n-k-1})$
2 $\widehat{\mathbf{w}} = (\widehat{w}_0, \widehat{w}_1, \ldots, \widehat{w}_{n-1}) \leftarrow \mathbf{y} - \widehat{\mathbf{z}} \cdot \mathbf{H}$
3 $\widehat{\mathbf{m}} \leftarrow (\widehat{w}_{n-k}, \widehat{w}_{n-k+1}, \ldots, \widehat{w}_{n-1})$
Output: message vector $\widehat{\mathbf{m}} \in [q]^k$

We are now ready to show a construction of u-PSMCs based upon the parity-check matrix of linear codes. The proof of its properties will be based on the result from Theorem 7.

Construction 4: Let $u \leq q + d - 3$, $u \leq n$, $k < n$, and let $\mathbf{H}$ be a systematic $(n-k) \times n$ parity-check matrix of an $[n, k, d]_q$ code. We define a u-PSMC by Algorithms 7 and 8.

The correctness and the parameters of the code from Construction 4 are proved in the next theorem.

Theorem 8: Let $u \leq q + d - 3$, $u \leq n$, $k < n$, and let $\mathbf{H}$ be a systematic $(n-k) \times n$ parity-check matrix of an $[n, k, d]_q$ code. Then, Construction 4 provides a u-PSMC over $\mathbb{F}_q$ of length n and redundancy $r = n - k$.

Proof: Since $\mathbf{H}$ is the parity-check matrix of an $[n, k, d]_q$ code, for $\ell \geq d - 1$, any $(n-k) \times \ell$ submatrix of $\mathbf{H}$ has rank at least $d-1$ and contains no all-zero column.

Let us consider the reduced row echelon (RRE) of any u columns of $\mathbf{H}$. If $u < d - 1$, it is a square $u \times u$ matrix of rank u and a special case of (3) with exactly one element in each "block". Else, it has at least $d-1$ rows and u columns of the following form:

$$\begin{pmatrix} 1 & \circ & \ldots & \ldots & \ldots & \circ\ \circ\ \ldots\ \circ \\ & 1 & \circ & \ldots & \ldots & \circ\ \circ\ \ldots\ \circ \\ & & 1 & \circ & \ldots & \circ\ \circ\ \ldots\ \circ \\ & & & \ddots & \ddots & \circ\ \circ\ \ldots\ \circ \\ 0 & & & & 1 & \underbrace{\circ\ \circ\ \ldots\ \circ}_{\leq u-d+1} \end{pmatrix}, \tag{5}$$

where $\circ$ is some element from $\mathbb{F}_q$. The matrix $\mathbf{H}$ contains no all-zero columns, and thus its RRE contains no all-zero columns. This holds since $\mathrm{RRE}(\mathbf{H}) = \mathbf{T}\cdot\mathbf{H}$ for some full-rank matrix $\mathbf{T}$ and therefore, for some column vector $\mathbf{b}^T$ (which denotes an arbitrary column of $\mathbf{H}$), the column vector $\mathbf{a}^T = \mathbf{T}\cdot\mathbf{b}^T$ (the corresponding column of $\mathrm{RRE}(\mathbf{H})$) is all-zero if and only if $\mathbf{b} = \mathbf{0}$.

Hence, each of the (at most) $u-d+1$ rightmost columns in (5) has at least one non-zero element. For any of these $u-d+1$ columns, its lowermost non-zero symbol determines to which of the first $d-1$ columns it will be associated. Hence, in the worst case, there is one such set of columns of size $u-d+2$ (the rightmost $u-d+1$ columns and one of the leftmost $d-1$ columns). Thus, for $u-d+2 \leq q-1$, (5) is a special case of (3). According to Theorem 7, we can therefore mask any u partially stuck-at-1 cells with this matrix and the redundancy is $r = n-k$.

Algorithm 7 shows the encoding procedure for this u-PSMC. For simplicity, a *systematic* $(n-k)\times n$ parity-check matrix $\mathbf{H}$ of the $[n,k,d]_q$ code is used.

The decoding is shown in Algorithm 8. Both algorithms work similarly as Algorithms 1 and 2, but the restriction on the error-correcting code $[n,k,d]_q$ is weaker and therefore our u-PSMC has a smaller redundancy than the u-SMC. ∎

Notice that for $q = 2$, a PSMC is just an SMC and the construction from Theorem 8 is equivalent to the one from Theorem 1. Theorem 8 leads to the following corollaries.

Corollary 1: If $q \geq 2$ is a prime power, then for all $u \leq n$, there exists a u-PSMC with redundancy $r = \max\{1, \rho_q(n, u-q+3)\}$ and therefore $r_q(n,u,\mathbf{1}) \leq \max\{1, \rho_q(n,u-q+3)\}$.

Corollary 2 (Construction 4 for $u=q$): If $q \geq 2$ is a prime power, then for all $u = q \leq n$, there exists a q-PSMC with redundancy $r = \rho_q(n,3)$ and therefore $r_q(n,q,\mathbf{1}) \leq \rho_q(n,3)$.

Theorem 8 shows that in many cases, we can decrease the redundancy by several q-ary symbols compared to SMCs. More specifically, according to Theorem 1, in order to construct codes which correct $u = q$ stuck-at cells, one needs to use a linear code with minimum distance $u+1$, while according to Theorem 8, for $u = q$ partially stuck-at-1 cells, it is enough to use a linear code with minimum distance 3. The next example demonstrates that in many cases, this improvement is quite large.

Example 5: Let $u = q = 5$ and $n = 30$. To mask u partially stuck-at-1 cells, we use the parity-check matrix of a $[30,27,3]_5$ code, which is the code of largest cardinality over $\mathbb{F}_5$ for $n = 30$ and $d = 3$ and thus its redundancy is $r = 3$ (see [10]).

To compare with SMCs, we need a code with minimum distance $d = 6$ (compare Theorem 1). The best known one according to [10] has parameters $[30,22,6]_5$ so its redundancy is $r = 8$. The trivial construction from the upper bound requires redundancy 4.16. Therefore, we improve upon both constructions.

However, note that it is still far from the lower bounds from (2) (which is 0.69) and from Theorem 3 (which is 1.11).

Another application of Construction 4 uses the parity-check matrix of the ternary Hamming code with parameters $[n = \frac{3^r-1}{2}, n-r, 3]_3$ to obtain a 3-PSMC. The redundancy of this construction is $r = \log_3(2n+1)$.

The following example shows how to use the same Hamming code, but double the code length and thus reduce the redundancy.

Example 6: Let $u = q = 3$, and $n = 8$. We will show how to encode 6 ternary symbols using the matrix

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 \end{pmatrix}.$$

Any two columns of $\left(\begin{smallmatrix}1&0&1&1\\0&1&1&2\end{smallmatrix}\right)$ are linearly independent (it is the parity-check matrix of the $[4,2,3]_3$ Hamming code), and therefore, any submatrix of $\mathbf{H}$, which consists of three columns, has rank two and its RRE is of the form stated in (3).

Let $\mathbf{m} = (m_0,m_1,m_2,m_3,m_4,m_5) = (1,0,2,0,1,2)$ be the message and assume that the cells at positions $0,2,4$ are partially stuck-at-1. We first set the memory state to be $\mathbf{w} = (0,0,m_0,m_1,m_2,m_3,m_4,m_5) = (0,0,1,0,2,0,1,2)$. Similar to Construction 2, we seek to find a length-2 ternary vector $\mathbf{z} = (z_0,z_1)$ such that the vector $\mathbf{y} = \mathbf{w} + \mathbf{z}\cdot\mathbf{H}$ masks the three partially stuck-at-1 cells. Hence, we can choose $\mathbf{z} = (1,1)$ and thus, $\mathbf{y} = (1,1,2,1,1,2,1,2)$ masks the three partially stuck-at-1 cells. It is possible to show that this property holds for any three partially stuck-at-1 cells and any message vector of length 6.

This provides a code of length $n = 8$ and redundancy $r = 2$ for masking any $u = 3$ partially stuck-at-1 cells. The first part of the upper bound from (1) uses the trivial construction and gives 2.95. The second part of the upper bound is $\rho_4(8,4) = 4$ (compare [10]) and thus, our construction requires less redundancy. The lower bound on the redundancy from (1) is 1.107, and the one from Theorem 3 is 1.161.

Example 6 is not a special case of Construction 4. The principle from Example 6 works for $u = 3$ and arbitrary q with the corresponding q-ary Hamming code. Unfortunately, it is not clear how to generalize it to arbitrary values of u or how to find other non-trivial matrices which fulfill the requirements of Theorem 7.

VI. 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 constructions from the previous sections achieve smaller redundancy, therefore and unless stated otherwise, we assume in this section that $n \geq u \geq q$.

For a vector $\mathbf{w} \in [q]^{n+1}$ 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 $\mathbf{w} \in [q]^{n+1}$ be given, then our construction of PSMCs has two main parts:

Algorithm 9 ENCODING-5$(\mathbf{m}; \mathbf{m}'; U)$

Input: • messages: $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1}) \in [q]^k$ and $\mathbf{m}' = (m'_0, m'_1, \ldots, m'_{n-k-2}) \in \left[\lfloor q/2 \rfloor\right]^{n-k-1}$
• positions of partially stuck-at-1 cells: $U = \{\phi_0, \phi_1, \ldots, \phi_{u-1}\} \subseteq [n+1]$

1 $\mathbf{w} = (w_0, w_1, \ldots, w_n) \leftarrow (\mathbf{0}_{n-k}, m_0, m_1, \ldots, m_{k-1}, 0)$
2 Find $z \in [q]$ such that $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$ entries that are equal to either zero or $(q-1)$, 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 $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in [2]^n$ be such that $v_i \leftarrow 1$ if $w_i^{(z)} = q-1$, else $v_i \leftarrow 0$
5 Apply Algorithm 1: $\mathbf{c}' \leftarrow \widetilde{\mathcal{E}}\big((v_{n-k}, v_{n-k+1}, \ldots, v_{n-1}); \widetilde{U}; \mathbf{1}_{\widetilde{u}}\big)$ $\widetilde{\mathbf{c}} \leftarrow (\mathbf{c}'\ 0)$
6 $\widetilde{\mathbf{m}} \leftarrow (2m'_0, 2m'_1, \ldots, 2m'_{n-k-2}, \mathbf{0}_{k+2}) \in [q]^{n+1}$
7 $\mathbf{y} \leftarrow (\mathbf{w}^{(z)} + \widetilde{\mathbf{c}} + \widetilde{\mathbf{m}}) \bmod q$

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

Algorithm 10 DECODING-5$(\mathbf{y})$

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 $\widehat{\mathbf{m}'} \leftarrow (\lfloor \widehat{y}_0/2 \rfloor, \lfloor \widehat{y}_1/2 \rfloor, \ldots, \lfloor \widehat{y}_{n-k-2}/2 \rfloor)$
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 $\widehat{\mathbf{m}'} \in \left[\lfloor q/2 \rfloor\right]^{n-k-1}$

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.

Construction 5: Let n, $q \geq 4$ and $u \leq n$ be positive integers and let $\widetilde{u} = \lfloor 2u/q \rfloor$. 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 Construction 1.

We define a u-PSMC of length $(n+1)$ over $[q]$ by Algorithms 9 and 10.

In the following theorem, we will prove the correctness of Construction 5 and afterwards give a detailed example to better illustrate the steps of the encoding and decoding algorithms.

Theorem 9: Let n, $q \geq 4$ and $u \leq n$ be positive integers and let $\widetilde{u} = \lfloor 2u/q \rfloor$. 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, Construction 5 provides 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 assume that $\mathbf{H}$ is a systematic $(n-k) \times n$ parity-check matrix of the code $\widetilde{\mathcal{C}}$ and we refer to Algorithms 1 and 2 as the encoder $\widetilde{\mathcal{E}}$ and decoder $\widetilde{\mathcal{D}}$ of the code $\widetilde{\mathcal{C}}$, respectively.

Let us start with the analysis of the **encoding algorithm**.

Step 2: we have to show that there exists $z \in [q]$ such that the vector $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$ entries which are equal to either zero or $(q-1)$. This 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

$$v_i = |\{j : w_j = i,\ j \in \{\phi_0, \ldots, \phi_{u-1}\}|,$$

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 $\sum_{i=0}^{q-1}(v_i + v_{(i+1) \bmod q}) = 2u$, according to 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 $\widetilde{u}$ entries that are equal to zero or $(q-1)$. The set of these positions is denoted by $\widetilde{U}$.

Step 3: we store the value of z in the last cell and if $z = 0$, we write $w_n = q-1$ which is necessary in the event where the *last* cell is partially stuck-at-1.

Steps 4 and 5: 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 $\widetilde{\mathcal{C}}$. In the first part of Step 5, we invoke Algorithm 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 $q-1$.

Note that only the last $n-k$ entries of $\mathbf{v}$ are passed to Algorithm 1, while the vector $\mathbf{v}$ corresponds to the vector $\mathbf{w}$ in Algorithm 1.

Steps 6 and 7: 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. 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 discuss the **decoding algorithm**.

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 $\widehat{z} = z$.

Step 2: with the value of $\widehat{z}$, we can reconstruct $\widehat{\mathbf{y}}$.

Step 3: we determine the value of the second message $\mathbf{m}'$. Since $\widehat{z} = z$, we get that $\widehat{\mathbf{m}'} = \mathbf{m}'$.

Step 4: we follow Algorithm 2 in order to decode the vector $\mathbf{c}'$ and get that $\widehat{\mathbf{c}'} = \mathbf{c}'$.

Step 5: we retrieve the first message vector and get $\widehat{\mathbf{m}} = \mathbf{m}$.

After proving the correctness of the encoding and decoding algorithms, we now calculate the redundancy. 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

$$\begin{aligned} r &= n + 1 - \log_q M = n + 1 - \log_q (q^k \cdot \lfloor q/2 \rfloor^{n-k-1}) \\ &= n - k + 1 - (n - k - 1) \cdot \log_q (\lfloor q/2 \rfloor) \\ &= (n - k - 1) \log_q \left(\frac{q}{\lfloor q/2 \rfloor} \right) + 2. \end{aligned}$$

■

The following example shows the complete encoding and decoding processes of Construction 5.

Example 7: 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 9 (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, $(\mathbf{w}^{(z)})_U = (1, 1, 3, 0, 1)$ and hence, $\widetilde{u} = 2$ and $\widetilde{U} = \{2, 3\}$. Further, we obtain:

$$\begin{aligned} &\textit{Step 4:} && \mathbf{v} = (0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1) \\ &\textit{Step 5:} && \mathbf{v}' = (0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1) \\ & && \mathbf{c}' = \widetilde{\mathcal{E}}(\mathbf{v}'; \{2, 3\}; (1, 1)) \\ & && \quad = (1, 0, 0, 0) \cdot \mathbf{H} \\ & && \quad = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1) \\ & && \widetilde{\mathbf{c}} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0) \\ &\textit{Step 6:} && \widetilde{\mathbf{m}} = (2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \\ &\textit{Step 7:} && \mathbf{y} = (0, \underline{1}, 3, 1, \underline{1}, 0, 3, 2, \underline{3}, 0, 1, 3, \underline{1}, 0, 0, \underline{1}) \end{aligned}$$

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

Let us now show the decoding process, i.e., Algorithm 10.

$$\begin{aligned} &\textit{Step 1:} && y_3 - y_{15} = 0 \Longrightarrow \widehat{z} = 1 \\ &\textit{Step 2:} && \widehat{\mathbf{y}} = (3, 0, 2, 0, 0, 3, 2, 1, 2, 3, 0, 2, 0, 3, 3, 0) \\ &\textit{Step 3:} && \widehat{\mathbf{m}'} = (\lfloor 3/2 \rfloor, 0, \lfloor 2/2 \rfloor) = (1, 0, 1) \\ &\textit{Step 4:} && \widehat{\mathbf{t}} = (3 - 2, 0 - 0, 2 - 2, 0) = (1, 0, 0, 0) \\ &\textit{Step 5:} && \widehat{\mathbf{c}'} = \widehat{\mathbf{t}} \cdot \mathbf{H} = (1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1) \\ &\textit{Step 6:} && \widehat{\mathbf{m}} = (0, 3, 2, 1, 2, 2, 3, 1, 3, 2, 2). \end{aligned}$$

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. Construction 4 from Corollary 1 requires redundancy $\rho_4(16, 4) = 4$.

The lower bound from (1) *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 8), but the principle of the construction is easier to show with short vectors.*

Example 8: Let $q = 4$, $n = 63$, $k = 57$ and $u = 5$. Then, the required redundancy for Construction 4 is $\rho_4(64, 4) = 6$ and for Construction 5, it 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 3: For all $4 \leq q \leq u \leq n$ we have that

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

Therefore, if we use a binary $[n = 2^{r_H} - 1, n - r_H, 3]_2$ Hamming code as a $\widetilde{u}$-SMC with $\widetilde{u} = 2$, then $u \leq q + \left\lfloor \frac{q-1}{2} \right\rfloor$ has to hold. For even q, we therefore have $u \leq q + q/2 - 1$. Then, the required redundancy is $r = (r_H - 1) \log_q(2) + 2 = (\log_2(n + 1) - 1) \cdot \log_q(2) + 2$.

Note that for $u \to n$ and for large q, the trivial construction from Section II is quite good. Construction 5 outperforms the trivial construction if $u > q$ and $n \gg u$.

VII. 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 set of positions which are partially stuck-at-i. Note that some values of u_i might be zero and thus the corresponding sets U_i are the empty set.

In the sequel of this section, we generalize the constructions from Sections IV, V and VI to the generalized model of partially stuck-at-$\mathbf{s}$ cells, where $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$. We only give the main theorems without showing the explicit decoding algorithms and mostly without examples

since these results are a direct consequence of the previous sections.

A. Generalization of Construction 2

Construction 2 from Theorem 4 (including its improvement from Theorem 5) generalizes as follows.

Theorem 10 (Generalized Construction 2): 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*

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

Proof: The proof is a generalization of the proofs of Theorem 4 and Theorem 5. Let $\mathbf{m} \in [q]^{n-1}$ be the message vector, define $\mathbf{w} = (0 \ \mathbf{m})$ and let $\mathbf{y} = \mathbf{w} + z \cdot \mathbf{1}_n$ be the vector that we will store. We have to find z such that $y_{\phi_i} = (w_{\phi_i} + z) \notin [s_i]$, $\forall i \in [u]$. Each partially stuck-at-s_i cell excludes at most s_i possible values for z, in total the u cells therefore exclude at most $\sum_{i=0}^{u-1} s_i$ values and if $\sum_{i=0}^{u-1} s_i < q$, there is at least one value $z \in [q]$ such that $y_{\Phi_i} = (w_{\Phi_i} + z) \notin [s_i]$, $\forall i \in [u]$.

In particular, there is always a $z \in [\sum_{i=0}^{u-1} s_i + 1]$ such that $y_{\phi_i} = (w_{\phi_i} + z) \notin [s_i]$, for all $i \in [u]$. Therefore, we can save additional information in the redundancy cell and as a generalization of Theorem 5, the statement follows. ∎

The encoding and decoding processes are analogous to Algorithms 3 and 4, and Algorithms 5 and 6, respectively. Unfortunately, we cannot claim (as in Theorem 6 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. For example, if $s_i = s$, $\forall i$, and $(su+1) \mid q$, then the redundancy of Theorem 10 is $r = \log_q(su+1)$, but the lower bound form Theorem 3 approaches only $r_q(n, u, s) = \log_q(u+1)$ for n large enough.

B. Generalization of Construction 4

In order to generalize Construction 4 from Theorem 8, let us first generalize Theorem 7.

Theorem 11: Let q *be a prime power and let a* $\kappa \times n$ *matrix* $\mathbf{H} = \left(H_{i,j}\right)_{j \in [n]}^{i \in [\kappa]}$ *over* $\mathbb{F}_q$ *be given. Let* $s = \max_{i \in [u]} s_i$.

If the RRE of any $\kappa \times u$ *submatrix (denoted by* $\mathbf{H}^{(u)}$*) has the following form (up to column permutations:)*

$$\mathrm{RRE}(\mathbf{H}^{(u)}) = \begin{pmatrix} \underbrace{1 \ \bullet \ \ldots \ \bullet}_{\leq \lceil q/s \rceil - 1} & \circ & \circ & \ldots & \circ & \ldots & \ldots & \ldots & \circ \\ & \underbrace{1 \ \bullet \ \ldots \ \bullet}_{\leq \lceil q/s \rceil - 1} & \circ & \circ & \ldots & \circ & \ldots & \ldots & \circ \\ & & \underbrace{1 \ \bullet \ \ldots \ \bullet}_{\leq \lceil q/s \rceil - 1} & \circ & \ldots & & \ldots & \ldots & \circ \\ & & & \underbrace{1 \ \bullet \ \ldots \ \bullet}_{\leq \lceil q/s \rceil - 1} & \ddots & & & \vdots & \\ & & 0 & & & & & \underbrace{1 \ \bullet \ \ldots \ \bullet}_{\leq \lceil q/s \rceil - 1} & \end{pmatrix}, \tag{6}$$

where $\bullet$ *has to be a non-zero element from* $\mathbb{F}_q$ *and* $\circ$ *is any element from* $\mathbb{F}_q$, *then, there exists a* (u, s)*-PSMC over* $[q]$ *of length* n *and redundancy* $r = \kappa$.

Proof: Follows from the proof of Theorem 7 with similar generalizations as in Theorem 10. ∎

Notice that the largest s_i restricts the size of the matrix and it is not clear how to consider the explicit values s_i in this generalization and not only the maximum value.

Based on Theorem 11, we can generalize Construction 4 as follows.

Theorem 12 (Generalized Construction 4): Given $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$. *Let* $s = \max_{i \in [u]} s_i$, *let* $u \leq \lceil q/s \rceil + d - 3$, $u \leq n$, $k < n$, *and let* $\mathbf{H}$ *be a systematic* $(n-k) \times n$ *parity-check matrix of an* $[n, k, d]_q$ *code. Then, there exists a* $(u, \mathbf{s})$*-PSMC over* $\mathbb{F}_q$ *of length* n *and redundancy* $r = n - k$.

Proof: Similar to the proof of Theorem 8 using Theorem 11. ∎

Corollary 4: If $q \geq 2$ *is a prime power, then for all* $u \leq n$ *and* $s \in [1, q-1]$, *there exists a* (u, s)*-PSMC with redundancy* $r = \max\{1, \rho_q(n, u - \lceil q/s \rceil + 3)\}$ *and therefore* $r_q(n, u, s) \leq \max\{1, \rho_q(n, u - \lceil q/s \rceil + 3)\}$.

C. Generalization of Construction 5

In the first step, we suggest a generalization to the case where all partially stuck-at cells are stuck at *the same level* s, for some $1 \leq s \leq q-1$. For this purpose, we use a Q-ary code, where $Q \geq s+1$ is a prime power, and the principle consists of the following two steps:

1) Find $z \in [q]$ such that the number of the values contained in the set $S = \{q-Q+1, \ldots, q-1, 0, \ldots, s-1\}$ of size $Q+s-1$ is minimized in $(\mathbf{w}^{(z)})_U = (\mathbf{w} + z \cdot \mathbf{1}_n)_U$. Denote the number of values from S in $(\mathbf{w}^{(z)})_U$ by $\widetilde{u}$.
2) Use a Q-ary $\widetilde{u}$-SMC to mask these $\widetilde{u}$ stuck cells.

This leads to the following theorem.

Theorem 13 (Generalized Construction 5 for $s_i = s$*): Let* n, $q \geq 4$, u *be positive integers, let* $s \in [1, q-1]$, *and let* $Q \geq s+1$ *be a prime power. Denote*

$$\widetilde{u} = \left\lfloor \frac{(Q+s-1)u}{q} \right\rfloor.$$

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, s)*-PSMC over* $[q]$ *of length* $(n+1)$ *and redundancy*

$$r = (n-k-1)\log_q\left(\frac{q}{\lfloor q/Q \rfloor}\right) + 2. \tag{7}$$

Proof: Let $S = \{q-Q+1, \ldots, q-1, 0, \ldots, s-1\}$ of cardinality $Q+s-1$. As in the proof of Theorem 9, we first show that there is some scalar z such that $(\mathbf{w}^{(z)})_U$ has at most $\widetilde{u}$ entries from S. The value $\widetilde{u}$ is equivalent to the minimum number of any $Q+s-1$ consecutive values in $\mathbf{w}_U$. Denote $v_i = |\{j : w_j = i, j \in \{\phi_0, \ldots, \phi_{u-1}\}|$, for all $i \in [q]$. Hence, we want to minimize $\sum_{j=0}^{Q+s-2} v_{i+j \bmod q}$, $i \in [q]$. Since $\sum_{i=0}^{q-1} \sum_{j=0}^{Q+s-2} v_{i+j \bmod q} = (Q+s-1)u$, there exists an integer i such that $\sum_{j=0}^{Q+s-2} v_{i+j \bmod q} \leq \lfloor \frac{(Q+s-1)u}{q} \rfloor$ and therefore, there is a z such that $(\mathbf{w}^{(z)})_U$ contains at most $\widetilde{u} = \lfloor \frac{(Q+s-1)u}{q} \rfloor$ values from S.

Second, we treat those $\widetilde{u}$ cells of $(\mathbf{w}^{(z)})_U$, which have values in S, like usual Q-ary stuck cells. Note that adding any value

from the set $[Q]$ to the other $u-\widetilde{u}$ cells still masks the partially stuck-at-s cells. Therefore, we can use an $[n, k, d \geq \widetilde{u}+1]_Q$ code to mask these cells (see Theorem 1).

As a generalization of Theorem 9, we choose $\mathbf{m} \in [q]^k$ and $\mathbf{m}' \in \left[\lfloor q/Q \rfloor\right]^{n-k-1}$ and the statement on the redundancy follows. ∎

Note that if $Q+s-1 \geq q$, then $\widetilde{u} \geq u$ and instead of using Theorem 13, we should use a q-ary u-SMC or another construction.

Let us now describe a method to generalize Construction 5 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. Of course, one way is to define $s=\max_i\{s_i\}$ and to consider the cells as partially stuck-at-s cells and use Theorem 13. However, we can do better by refining the "minimization step" as done in the following theorem.

Theorem 14 (Generalized Construction 5): 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

$$r=(n-k-1)\log_q\left(\frac{q}{\lfloor q/Q \rfloor}\right)+2. \tag{8}$$

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 $i \in [1, q-1]$.

In the sequel, we describe a way how accomplish this. We have

$$\sigma_i \stackrel{\text{def}}{=} \left|[q] \setminus [i, q-Q]\right| = \min\{q, Q+i-1\}, \quad \forall i \in [q],$$

and generalizing Theorems 9 and 13, we define:

$$v_\ell^{(i)}=|\{j : w_j=\ell, j \in U_i\}|, \quad \forall \ell \in [q],\ i \in [1, q-1]. \tag{9}$$

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

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

Similar to Theorems 9 and 13, we can treat these $\widetilde{u}$ partially stuck-at cells as Q-ary cells which are usually stuck-at. With a similar proof as in Theorems 9 and 13, this leads to the statement. ∎

Example 9: Let $n=31$, $q=8$, and $\mathbf{s}=(1\ 1\ 1\ 1\ 2\ 2\ 3)$. Hence, $u_1=4$, $u_2=2$, $u_3=1$ and $u=7$. We choose $Q=4$ and we want to use a 4-ary SMC to construct a $(u, \mathbf{s})$-PSMC of length 32.

The first approach is to consider all cells as partially stuck-at level $s=\max_i\{s_i\}=3$. Then, with Theorem 13 and $Q=4$, we have $\widetilde{u}=5$ and we need a code over $\mathbb{F}_{2^2}$ of length 31 and distance at least 6. The largest known such code is a $[31, 23, 6]_4$ code (see [10]) and therefore, with (7), the required redundancy is 6.66.

Let us now compare this to Theorem 14. We denote $\sigma_1=4$, $\sigma_2=5$ and $\sigma_3=6$ and therefore,

$$\widetilde{u}=\left\lfloor \frac{4u_1+5u_2+6u_3}{q} \right\rfloor = 4.$$

We need a code over $\mathbb{F}_{2^2}$ of length 31 and distance at least $\widetilde{u}+1=5$. The largest such known code is a $[31, 24, 5]_4$ code (see [10]) and with (8), the required redundancy is $r=6$, i.e., compared to the approach of Theorem 13, we have decreased the redundancy.

The generalized Construction 4 from Theorem 12 needs redundancy $\rho_8(32, 7)=10$. Further, the upper bound from Theorem 2, gives 7.01 and we also improve upon this bound.

VIII. 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. [8]. 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.,*

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

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

The following theorem establishes a connection between PSMCs and UMCs.

Theorem 15: Given 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 $\overline{\mathbf{y}^{(m)}} \stackrel{\text{def}}{=} (q-1-y_0^{(m)}, q-1-y_1^{(m)}, \ldots, q-1-y_{n-1}^{(m)})$ 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 $\widetilde{\mathbf{s}} = (s_0, s_1, \ldots, s_{u-1})$ and redundancy r. ∎

Notice that the converse statement also holds, i.e., a $(u, \widetilde{\mathbf{s}})$-UMC with $\widetilde{\mathbf{s}} = (s_0, s_1, \ldots, s_{u-1})$ and redundancy r can be used as 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.

The construction by Gabrys et al. from [8] is based on tensor product codes and can mask unreliable cells and correct additional random errors. To compare their construction to our results, we assume that no random errors occur (i.e., in their notation $t_1 = t_2 = 0$). Then, their matrix H_3 defines an $[n, n, 0]_4$ code (in their notation, meaning it does not correct any error), H_4 defines an $[n, k_4, 0, \ell]_2$ code (meaning it is a ℓ-SMC which cannot correct additional random errors). Therefore, $d_4 \geq \ell + 1$, see e.g., [11] and due to the Singleton bound $k_4 \leq n - d_4 + 1 = n - \ell$. The construction from [8, p. 1492] yields a code for masking the unreliable cells of length $3n$ and dimension $2k_3 + k_4 \leq 3n - \ell$. Thus, their required redundancy is $r \geq \ell$, which is the same as the one for a ℓ-SMC and thus, their construction does not give any improvement compared to [11]. As shown in the previous sections and in Theorem 15, our constructions therefore improve on both, [8] and [11], when masking memory cells with unreachable levels.

IX. 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 by Heegard and El Gamal in [12]. 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

$$C_q(p, s) = 1 - p + p \log_q(q-s) = 1 - p \log_q\left(\frac{q}{q-s}\right). \tag{10}$$

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. Furthermore, according to [12], 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, we conclude that (10) 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 VI 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 9 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 rate of this construction is given by

$$C_q(p, s) - R_q(p, s) = p \cdot \underbrace{\left(\frac{2s}{q} \log_q\left(\frac{q}{\lfloor q/(s+1) \rfloor}\right) - \log_q\left(\frac{q}{q-s}\right)\right)}_{\stackrel{\text{def}}{=} \Delta(q,s)}, \tag{11}$$

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

Table I shows the values of the difference coefficient $\Delta(q, s)$ for different values of q and s.

In the same way, 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 (11). A similar analysis can be done for the $(u, \mathbf{s})$-PSMC construction from Theorem 14.

For the special case $s = 1$ and when q is large enough, we neglect the floor operation and we can use the approximation $\ln(q) - \ln(q-1) \approx 1/q$ and obtain

$$\begin{aligned} C_q(p, 1) - R_q(p, 1) &= \\ &\approx p\left(\frac{2\log_q(2)}{q} - \log_q\left(\frac{q}{q-1}\right)\right) \\ &\approx p\left(\frac{2\ln(2)}{q\ln(q)} - \frac{1}{q\ln(q)}\right) \approx 0.38\frac{p}{q\ln(q)}. \end{aligned} \tag{12}$$

TABLE I

THE DIFFERENCE COEFFICIENT $\Delta(q,s)$ FOR DIFFERENT VALUES OF q AND s, WHERE $0 < s < q$ WHERE $s+1$ IS A PRIME POWER

	$s=1$	$s=2$	$s=3$	$s=4$	$s=6$	$s=7$
$q=2$	0					
$q=3$	0.29	0.33				
$q=4$	0.042	0.5	0.5			
$q=5$	0.089	0.48	0.63	0.6		
$q=6$	0.027	0.18	0.61	0.72		
$q=7$	0.045	0.19	0.57	0.71	0.71	
$q=8$	0.019	0.19	0.27	0.67	0.83	0.75
$q=11$	0.020	0.11	0.25	0.33	0.76	0.85
$q=13$	0.015	0.076	0.16	0.31	0.68	0.77
$q=16$	0.0079	0.057	0.11	0.19	0.39	0.45
$q=21$	0.0072	0.036	0.084	0.14	0.25	0.38
$q=32$	0.0033	0.023	0.047	0.082	0.17	0.19

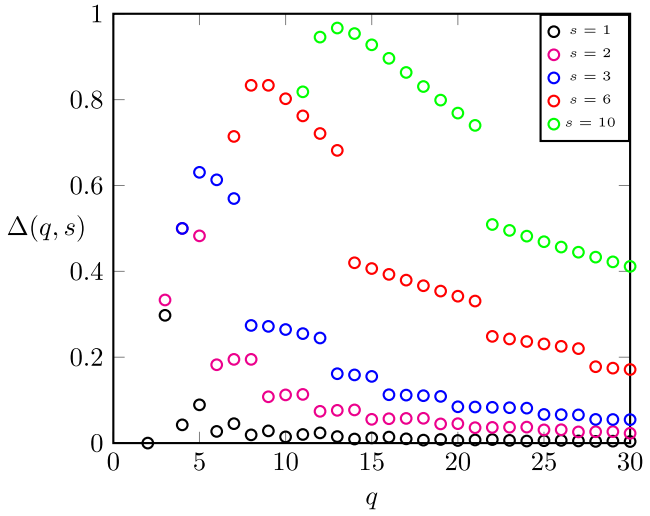

Fig. 1. The difference coefficient $\Delta(q,s)$ for different values of q and s.

Thus, we can evaluate how close the code rate from Theorems 9 and 13 come to the capacity from (10). Further, (12) shows that asymptotically, the coefficient $\Delta(q,1)$ approaches zero when q increases. In the same way, for fixed s, the coefficient $\Delta(q,s)$ approaches 0 with increasing q.

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 $1/(2\ln 2)\approx 0.7213$.

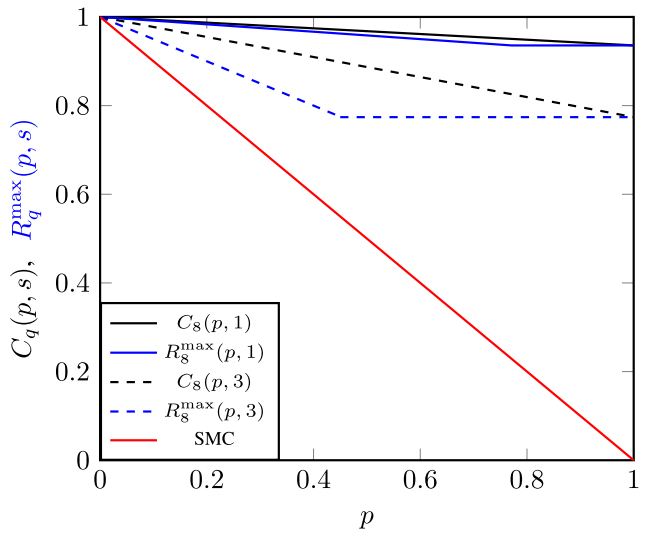

Fig. 2. Capacity $C_q(p,s)$ and maximum achievable rate $R_q^{\max}(p,s)$ for $q=8$.

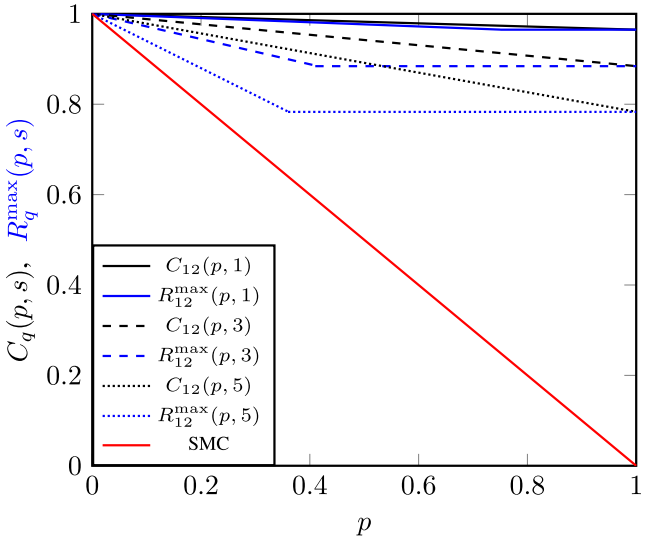

Fig. 3. Capacity $C_q(p,s)$ and maximum achievable rate $R_q^{\max}(p,s)$ for $q=12$.

We conclude that the maximum achievable rates, denoted by $R_q^{\max}(p,s)$, according to the constructions from Theorems 2, 9 and 13, 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 Figures 2 and 3, we plot the graphs of the capacity $C(p,s)$ and the maximum achievable rates $R_q^{\max}(p,s)$ for $q=8$, $s=1,3$ and $q=12$, $s=1,3,5$.

Notice that asymptotically, both an SMC and Construction 4 have rate $R=1-p$ and are therefore worse than $R_q^{\max}(p,s)$.

Lastly, we mention that there is a strong connection between codes for stuck-at cells and two-write *write-once memory* (WOM) codes. WOM codes are used to write information in memories whose cells can only increase their

levels [24]. In general, a binary u-SMC can be used to construct a two-write WOM code simply by treating the stuck-at cells as the ones which were programmed on the first write. Furthermore, a capacity achieving code for binary stuck-at cells will generate a capacity achieving two-write WOM codes. However, the converse does not necessarily hold. Even though constructions of WOM codes usually restrict the maximum number of cells that can be programmed on the first write, not necessarily all patterns with at most this maximum number of programmed cells are allowed. However, still several constructions of such WOM codes can be used to construct u-SMC, see e.g. [26], [27], [31].

The generalization to the non-binary case is very similar in the sense that PSMCs can be used to construct non-binary two-write WOM codes. We simply let the partially stuck-at cells be the ones which are programmed on the first write and then use the PSMC to be the encoder and decoder for the second write of the WOM code.

In [5], Burshtein and Strugatski presented a construction of polar WOM codes both for the binary and non-binary cases. Recently, Burshtein extended these results in [4] to an arbitrary side information channel. In [22], Mahdavifar and Vardy showed another construction of codes for stuck-at cells which are based on polar codes and LDPC codes. We believe that these constructions can be adapted to derive capacity achieving PSMCs in our model. However, note that these constructions are not zero-error codes as the ones we propose in this paper. The extension of these families of codes to PSMCs is an interesting problem for future research.

X. 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 three 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., $s = 1$) and $u+1$ divides q, this construction is asymptotically optimal. The second construction is based on the parity-check matrix of an error-correcting code and works well for $u \geq q$ (where q is a prime power). The last construction is based on using a binary error-correcting code and can be applied for any u and q. The three 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.

The following table for $s = 1$ provides an overview of our constructions and their required redundancies. Recall that $\rho_q(n, d)$ denotes the smallest redundancy of a linear error-correcting code of length n and minimum Hamming distance d over $\mathbb{F}_q$.

	Upper bound on $r_q(n, u, \mathbf{1})$
$u < q$	$1 - \log_q \left\lfloor \frac{q}{u+1} \right\rfloor$ (Theorem 4)
$u \geq q$	$\rho_q(n, u - q + 3)$ (Corollary 1)
any u, q	$\left(\rho_2\left(n-1, \lfloor \frac{2u}{q} \rfloor + 1\right) - 1\right) \cdot \log_q\left(\frac{q}{\lfloor q/2 \rfloor}\right) + 2$ (Theorem 9)

Similarly, the following table gives an overview of the constructions for arbitrary partially stuck-at levels, given by a vector $\mathbf{s} = (s_0, s_1, \ldots, s_{u-1}) \in [1, q-1]^u$.

	Upper bound on $r_q(n, u, \mathbf{s})$
$\sum_{i=0}^{u-1} s_i < q$	$r = 1 - \log_q \left\lfloor \frac{q}{\sum_{i=0}^{u-1} s_i + 1} \right\rfloor$ (Theorem 10)
$u \geq \left\lceil \frac{q}{\max_i s_i} \right\rceil$	$\rho_q\left(n, u - \left\lceil \frac{q}{\max_i s_i} \right\rceil + 3\right)$ (Corollary 4)
any u, q	$\left(\rho_Q\left(n-1, \left\lfloor \frac{\sum_{i=1}^{q-1} u_i \cdot \sigma_i}{2} \right\rfloor + 1\right) - 1\right) \cdot \log_q\left(\frac{q}{\lfloor q/Q \rfloor}\right) + 2,$ where $Q \geq \max_i\{s_i\}+1$ is a prime power and $\sigma_i = \min\{q, Q + i - 1\}$, $\forall i \in [q]$. (Theorem 14)

Thus, we have provided codes for masking partially stuck-at cells for any set of parameters and have shown that the required redundancy significantly decreases compared to SMCs and to a trivial construction.

Acknowledgement

The authors would like to thank the associate editor Moshe Schwartz and the anonymous reviewers for their very helpful comments.

References

[1] I. Belov and A. M. Shashin, "Codes that correct triple defects in memory," (in Russian) *Problems Inf. Transmiss.*, vol. 13, no. 4, pp. 62–65, 1977.

[2] J. Borden and A. J. Vinck, "On coding for 'stuck-at' defects (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 33, no. 5, pp. 729–735, Sep. 1987.

[3] G. W. Burr *et al.*, "Phase change memory technology," *J. Vac. Sci. Technol. B*, vol. 28, no. 2, pp. 223–262, 2010.

[4] D. Burshtein, "Coding for asymmetric side information channels with applications to polar codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 1527–1531.

[5] D. Burshtein and A. Strugatski, "Polar write once memory codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 5088–5101, Aug. 2013.

[6] C. L. Chen, "Linear codes for masking memory defects (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 31, no. 1, pp. 105–106, Jan. 1985.

[7] J. Fridrich, M. Goljan, and D. Soukal, "Steganography via codes for memory with defective cells," in *Proc. Allerton Conf. Commun., Control Comput.*, Sep. 2005, pp. 1–17.

[8] R. Gabrys, F. Sala, and L. Dolecek, "Coding for unreliable flash memory cells," *IEEE Commun. Lett.*, vol. 18, no. 9, pp. 1491–1494, Sep. 2014.

[9] B. Gleixner, F. Pellizzer, and R. Bez, "Reliability characterization of phase change memory," in *Proc. 10th Annu. Non-Volatile Memory Technol. Symp. (NVMTS)*, Oct. 2009, pp. 7–11.
[10] M. Grassl. (2007). "Bounds on the minimum distance of linear codes and quantum codes." [Online]. Available: http://www.codetables.de
[11] C. Heegard, "Partitioned linear block codes for computer memory with 'stuck-at' defects," *IEEE Trans. Inf. Theory*, vol. 29, no. 6, pp. 831–842, Nov. 1983.
[12] C. Heegard and A. El Gamal, "On the capacity of computer memory with defects," *IEEE Trans. Inf. Theory*, vol. 29, no. 5, pp. 731–739, Sep. 1983.
[13] K. Kim and S. Ahn, "Reliability investigations for manufacturable high density PRAM," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2005, pp. 157–162.
[14] Y. Kim and B. V. K. V. Kumar, "Coding for memory with stuck-at defects," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2013, pp. 4347–4352.
[15] A. V. Kuznetsov, T. Kasami, and S. Yamamura, "An error correcting scheme for defective memory," *IEEE Trans. Inf. Theory*, vol. 24, no. 6, pp. 712–718, Nov. 1978.
[16] A. Kuznetsov, "Coding in a channel with generalized defects and random errors," (in Russian) *Problems Inf. Transmiss.*, vol. 21, no. 1, pp. 28–34, 1985.
[17] A. Kuznetsov and B. Tsybakov, "Coding for memories with defective cells," (in Russian) *Problems Inf. Transmiss.*, vol. 10, no. 2, pp. 52–60, 1974.
[18] L. A. Lastras-Montaño, A. Jagmohan, and M. M. Franceschini, "Algorithms for memories with stuck cells," in *Proc. IEEE Int. Symb. Inf. Theory*, Jun. 2010, pp. 968–972.
[19] L. Lastras-Montaño, A. Jagmohan, and M. M. Franceschini, "An area and latency assessment for coding for memories with stuck cells," in *Proc. IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2010, pp. 1851–1855.
[20] 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 Lett.*, vol. 30, no. 5, pp. 448–450, May 2009.
[21] V. V. Losev, V. K. Konopel'ko, and Y. D. Daryakin, "Double-and-triple-defect-correcting codes," (in Russian) *Problems Inf. Transmiss.*, vol. 14, no. 4, pp. 98–101, 1978.
[22] H. Mahdavifar and A. Vardy, "Explicit capacity achieving codes for defective memories," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 641–645.
[23] A. Pirovano *et al.*, "Reliability study of phase-change nonvolatile memories," *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 422–427, Sep. 2004.
[24] R. L. Rivest and A. Shamir, "How to reuse a write-once memory," *Inf. Control*, vol. 55, nos. 1–3, pp. 1–19, Dec. 1982.
[25] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "SAFER: Stuck-at-fault error recovery for memories," in *Proc. MICRO*, Dec. 2010, pp. 115–124.
[26] A. Shpilka, "New constructions of WOM codes using the Wozencraft ensemble," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4520–4529, Jul. 2013.
[27] A. Shpilka, "Capacity-achieving multiwrite WOM codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 3, pp. 1481–1487, Mar. 2014.
[28] B. S. Tsybakov, S. I. Gelfand, A. V. Kuznetsov, and S. I. Ortyukov, "Reliable computation and reliable storage of information," in *Proc. IEEE-USSR Workshop*, Dec. 1975.
[29] B. S. Tsybakov, "Defects and error correction," (in Russian) *Problems Inf. Transmiss.*, vol. 11, no. 1, pp. 21–30, 1975.
[30] B. S. Tsybakov, "Group additive defect-correcting codes," (in Russian) *Problems Inf. Transmiss.*, vol. 11, no. 1, pp. 111–113, 1975.
[31] Y. Wu, "Low complexity codes for writing a write-once memory twice," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2010, pp. 1928–1932.

Antonia Wachter-Zeh (S'10–M'14) received a B.S. degree in electrical engineering in 2007 from the University of Applied Science Ravensburg, Germany, and the M.S. degree in communications technology in 2009 from Ulm University, Germany. She obtained her Ph.D. degree in 2013 at the Institute of Communications Engineering, University of Ulm, Germany and at the Institut de recherche mathématique de Rennes (IRMAR), Université de Rennes 1, Rennes, France. Currently, she is a postdoctoral researcher at the Technion- Israel Institute of Technology, Haifa, Israel. Her research interests are coding and information theory and their applications.

Eitan Yaakobi (S'07–M'12) is an Assistant Professor at the Computer Science Department at the Technion - Israel Institute of Technology. He received the B.A. degrees in computer science and mathematics, and the M.Sc. degree in computer science from the Technion - Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California, San Diego, in 2011. Between 2011-2013, he was a postdoctoral researcher in the department of Electrical Engineering at the California Institute of Technology. His research interests include information and coding theory with applications to non-volatile memories, associative memories, data storage and retrieval, and voting theory. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010-2011.