Implementation of Stuck-at Cells

Abstract - In this paper we will provide an implementation of the Stuck-at Cell coding scheme. The input of our program is a binary matrix of a parity check matrix of size r × n for a length-n code with minimum Hamming distance d.
We will use this matrix in order to mask any d − 1 stuck at cells. The solution includes both the encoding and decoding maps and will be based on the construction from [2] and [4, Const. 1] .

I. INTRODUCTION

Phase-Change Memory (PCM) is a form of computer random-access memory (RAM) that stores data by altering the state of the matter from which the device is fabricated. PCM has multiple distinct physical states; one amorphous and some partially crystalline.

As a result of this alteration process we can exhibit two types of imperfections that impede the data reliability. The first type is a defective memory cell, whose cell value is stuck-at a particular value independent of the input. For example, some of the cells of a binary memory may be stuck-at 0, and when a 1 is attempted to be written into a stuck-at 0 cell, an error results. The second type of imperfection is a noisy cell which can occasionally result in a random error.

This paper shall address the former imperfection, the defective memory cell problem. Using the classical version of stuck-at cells (see e.g. [3]), in which the memory consists of n binary cells of which u are stuck-at either in the zero or one state.

A distinction can be made - between Partially stuck-at cells, and stuck-at cells.
Partially stuck-at cells are a problem in which a cell may not be stuck on a definite and final value - but it may be stuck at a minimum level s belonging to {0,1,..,q-1} (noted [q]) - that is, it can accept any level $\geqslant s$.
On the other hand, Stuck-at cells, is a problem in which a cell is stuck at a finite and fixed level $s \in [q]$.

The problem in hand is the implementation of the Stuck-at Cell coding scheme. A code for stuck-at cells is needed. This code maps message vectors on code-words which mask the stuck-at cells, i.e., the values of each code-word at the stuck positions match to their stuck level. The encoder knows the locations and values of the stuck-at cells, while the decoder does not. Thus, 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 subsequently successfully decoded.

The origin of this problem is a paper by Kusnetsov and Tsybakov [3]. They consider coding for binary memories that have a fixed fraction p of stuck-at cells. The assumption is that the location and nature of the defects are available to the encoder and not to the decoder.

In this paper, we propose a simple scheme for Encoding and Decoding on a PCM containing stuck-at cells. Focusing on the case in which the locations and the stuck at values are known during encoding, whilst decoding should be carried out in the absence of this information.

The rest of the paper is as follows. Section **II** explains the channel model. In section **III**, Notations and Definitions for our coding scheme are introduced. In section **IV**, the algorithm for our encoding and decoding scheme is proposed, with relevant proofs. Section **V** concludes the paper.

II. THE CHANNEL MODEL

We start by introducing the communication system [1]. This system describes the information exchange between two points. The process of transmission and reception of information is called communication. The major elements of communication are The Information Transmitter, The Channel, and The Information Receiver.

Figure 1.1 shows a communication system for transmitting information from a source to a destination through a channel.

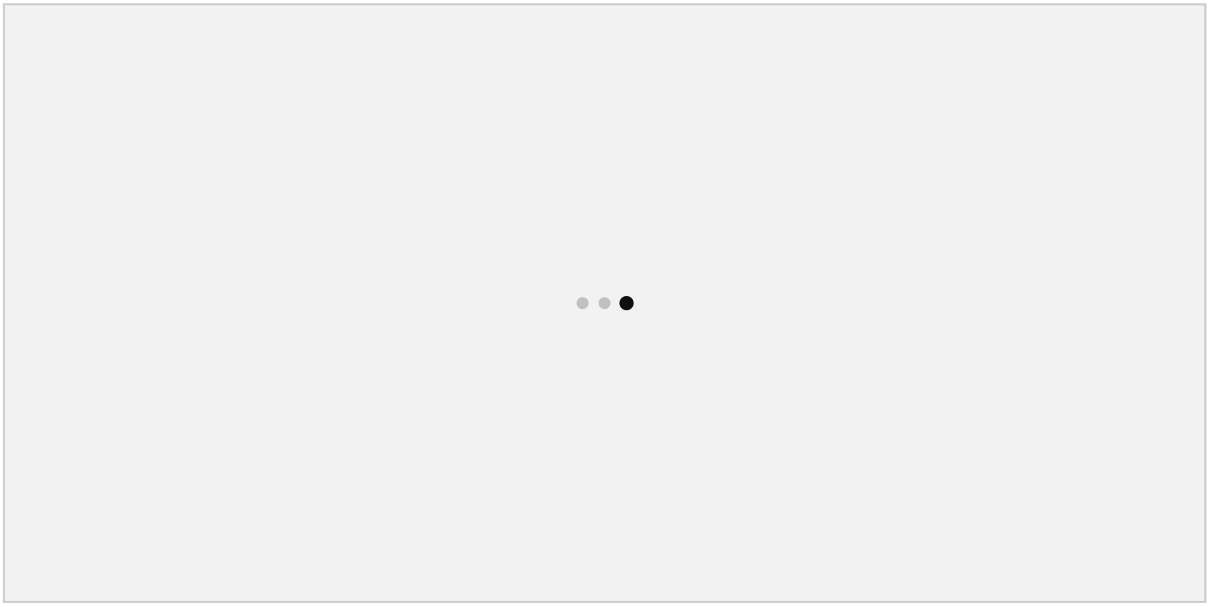

Figure 1

The transmitter has information that it wants to transmit to a receiver. Any discrete message can be represented with a string of binary numbers, so the transmitter uses a source encoder to encode the information message as a binary sequence. The transmitter can also add error-correcting codes using the channel encoder block, which essentially introduces redundancy by adding extra bits. The receiver can use the added redundancy to mitigate the errors that may be introduced by the channel. Finally, the transmitter must modulate the channel symbols (i.e., the output of the channel encoder) onto a carrier signal and release the signal for propagation in the channel.

Channel coding

We will concentrate on the channel coding part of Figure 1, as shown in Figure 2.

Figure 2

Usually, the channel introduces errors in the transmitted data. Typical errors can be substitution of symbols, insertions, deletions, erasures and more. But in this paper we shall only focus on the stuck-at problem.

Nevertheless, we define the process above as follows, assuming we use an (n, M) code C to transmit over a channel. The steps required are as follows:

1. An encoder to the code C is a mapping EC : {1, . . . , M} → C which maps the user information u into a codeword c ∈ C. Typically, when M is a power of 2 and k = log2 M the encoder can be denoted by Ec: $\{0,\ 1\}^k \rightarrow C$, so the user input is a word of k bits.
2. Given the user information u, its encoded codeword c = Ec(u) is transmitted over the channel, and the resulting channel's output is a noisy word y ∈ $\{0,\ 1\}^n$ which is fed to a channel decoder.
3. A decoder to the code C is a mapping Dc : $\{0,\ 1\}^n \rightarrow C$, which receives the channel's output y and outputs a codeword c (and thus also the information u), with the aim of having $\bar{c}$ = c and $\bar{u}$ = u.

III. NOTATIONS AND DEFINITIONS

Notations

n: code length.
k: code dimension.
d: minimal hamming distance.
r: code redundancy = n-k.
[n]: the set of integers {0,1,..,n-1}.

Definitions [5]

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], i.e., code of length n over the alphabet [q] with M codewords. Its redundancy is denoted by r = n− $log_q(M)$. A linear code over F_q , will be denoted by $[n, d, k]_q$, where k is it's dimension, it's redundancy is r = n − k, and d refers to the minimum Hamming distance of the code. ρ_q (n, d) is the smallest redundancy of any linear code of length n and minimum Hamming distance d over F_q . Codes for (partially) stuck-at cells are defined as follows.

An $(n, M)_q$ u-stuck-at-masking code (u-SMC) C is a coding scheme with encoder E and decoder D. The input to the encoder E is the set of locations $\{\varphi_0, \varphi_1, ..., \varphi_{u-1}\} \subseteq [n]$, the stuck levels $\{s_0, s_1, ..., s_{u-1}\} \subseteq [q]$ of some $u \leq n$ stuck-at cells and a message m ∈ [M]. Its output is a vector $y^{(m)} \in [q]^n$ which matches the values of the u stuck-at cells, i.e., $y^{(m)}_{\varphi_i} = s_i, \ \forall i \in [u]$ and its decoded value is m, that is D($y^{(m)}$) = m.

IV. THE ALGORITHM

Algorithm 1 Encoding $(m;\ H;\ \varphi_0, \varphi_1, ..., \varphi_{u-1}; s_0, s_1, ..., s_{u-1})$

Input:

- message: m = $(m_0, m_1, ..., m_{k-1}) \in \{0,1\}^k$
- binary matrix of a parity check matrix of size r × n: H
- positions of stuck-at cells: $\{\varphi_0, \varphi_1, ..., \varphi_{u-1}\} \subseteq [n]$
- levels of stuck-at cells: $\{s_0, s_1, ..., s_{u-1}\} \subseteq [q]$

1. $w = \{w_0, w_1, ..., w_{n-1}\} \leftarrow (0_{n-k}, m_0, m_1, ..., m_{k-1})$
2. find $z \in \{0,1\}^{n-k}$ s.t. $w + z * H = \overline{0}$
3. choose a vector z_0 which maintains the above

Output: vector $y = w + z_0 * H$

Algorithm 2 Decoding $(y;\ H;\ \varphi_0, \varphi_1, ..., \varphi_{u-1}; s_0, s_1, ..., s_{u-1})$

Input:

- coded message: y
- binary matrix of a parity check matrix of size r × n: H
- positions of stuck-at cells: $\{\varphi_0, \varphi_1, ..., \varphi_{u-1}\} \subseteq [n]$
- levels of stuck-at cells: $\{s_0, s_1, ..., s_{u-1}\} \subseteq [q]$

1. For $y = \{y_0, y_1, ..., y_{n-1}\}$, we take $z = \{y_0, y_1, ..., y_{n-k}\}$.
2. Get $last = \{x_0, x_1, ..., x_{n-k}\}$ for $x = y - z * H$
3. Extract $m = \{x_{n-k}, ..., x_n\}$ for the above last

Output: message vector $m \in \{0,1\}^k$

The above algorithm uses the [n,k,u+1] Hamming code , relying on the theorem proposed by Heegard in 1985: An [n,k,u+1] code can mask u stack-at cells.

We now prove the correctness of our algorithm.

Theorem:

Our coding scheme can be used to encode and decode up to d-1 stuck at cells.

Proof:

Encoding:

We are given: a message: m = $(m_0, m_1, ..., m_{k-1}) \in \{0,1\}^k$ a binary matrix of a parity check matrix of size r × n: H, the positions of stuck-at cells: $\{\varphi_0, \varphi_1, ..., \varphi_{u-1}\} \subseteq [n]$ and levels of stuck-at cells: $\{s_0, s_1, ..., s_{u-1}\} \subseteq [n]$

We begin by defining w, $w = \{w_0, w_1, ..., w_{n-1}\} \leftarrow (0_{n-k}, m_0, m_1, ..., m_{k-1})$.

We are searching for a vector $z \in \{0,1\}^{n-k}$ s.t. $w + z * H = \overline{0}$.

The above choice of $\bar{z}$ eventually matches the stuck cell values (z corresponds to the linear combination of H rows added to w).
Denote $z \cdot H = x$ we want the following to stand:

$$\forall 0 \le j \le u-1 : \; w_{ij} + x_{ij} = s_{ij}$$
$$\Rightarrow \forall 0 \le j \le u-1 : \; w_{ij} + (z \cdot H)_{ij} = s_{ij}$$

Denote by $I = \{i_0, i_1, ..., i_{u-1}\}$ The stuck-at index group.
We want to solve the equation:

$$z \cdot H_I = (s_0 - w_{i0},\ s_1 - w_{i1}, ...,\ s_{u-1} - w_{i(u-1)})$$

Because H_I isnt of a full rank, it is not invertible - thus we solve it with the least squares technique on the transposed equation:
$H^T * z^T = x^T$, which enables us to solve this LS problem:
$H * H^T * z^T = H * x^T$, where we know that $H * H^T$ is of full rank, and thus, invertible.
There is a single solution for the above LS problem, which is a solution for z^T .

We will mark this solution with z_0 and therefore write to memory:

$$y = w + {z_0}^T \cdot H$$

As z_0 corresponds to the transposed original equation.

Decoding:
We will now see that we will receive the exact same message that was given to us.

Using the y code-word that was given to us by the encoder, we start by picking the first n-k bits from it: We take $z = \{y_0, y_1, ..., y_{n-k-1}\}$ from the given y.

This is the exact same ${z_0}^T$ from the encoding step. (This is due to the construction of w with n-k zeroes at the start of w).
We then calculate $x = y - z * H$, which eventually returns us w from the encoding step.

By the construction of w in the encoding step, if we take $m = \{x_{n-k}, ..., x_n\}$, we will receive the exact same message that was encoded to y.

Thus, our scheme can satisfy and solve the problem of d-1 stuck-at cells.

V. CONCLUSION

In this paper, we have considered codes for masking partially stuck-at memory cells.
After defining the channel model and stuck-at cells, we presented our construction of an encoder/decoder which is capable of handling the defective cells problem.
The final redundancy of our encoder/decoder is n-k, as we use code-words from the length of n-k + k = n where our message is of length k.

ACKNOWLEDGEMENT

Would like to thank Associate Professor Eitan Yaakobi (Course Instructor) and Daniella Bar-Lev (Teaching Assistant).

APPENDIX

Below is the “readme” for the implementation of our algorithm:

The main.py file contains the encoder/decoder code for the stuck-at cells problem.

The folder currently includes an example for a channel which can receive messages of length 4.

There is a Parity Checking Matrix (pcm) attached and an input file for the indices of the stuck cells.

To use the code scheme:

1. Attach the desired parity checking matrix in an excel file format, with the name of: "pcm". There is an example pcm attached to this folder.
2. Attach the indices of the stuck cells in the file named "input" (also an excel file).
3. Run the python file, and input messages in the following format:
   To input a message to the memory, follow the instructions written on the screen, and enter "e" to encode a message to the memory.
   Enter a message of length of 4, for a pcm with n=7, using the format: "xxxx", for example.
4. To receive a message that was saved to the memory, enter the decoder by passing "d" to the input line and then enter the number of the message you would like to retrieve.

BIBLIOGRAPHY

[1] Farsad, Nariman & Guo, Weisi & Eckford, Andrew. (2013). Tabletop Molecular Communication: Text Messages through Chemical Signals. PloS one. 8. e82935. 10.1371/journal.pone.0082935.

[2] C. Heegard, “Partitioned linear block codes for computer memory with ‘stuck-at’ defects,” IEEE Transactions on Information Theory, vol. 29, no. 6, pp. 831–842, Nov. 1983.

[3] A. Kuznetsov and B. Tsybakov, “Coding for memories with defective cells,” (in Russian) Problems Inf. Transmiss., vol. 10, no. 2, pp. 52–60, 1974.

[4] A. Wachter-Zeh and E. Yaakobi, “Codes for partially stuck-at memory cells,” IEEE Transactions on Information Theory, vol. 62, no. 2, pp. 639–654, Feb. 2016.

[5] Tutorial 7 - webcourse.