Constructions for Constant-Weight ICI-Free Codes

Scott Kayser and Paul H. Siegel
University of California, San Diego, La Jolla, CA 92093, USA
Email: {skayser, psiegel}@ucsd.edu

Abstract—**In this paper, we consider the joint coding constraint of forbidding the 101 subsequence and requiring all codewords to have the same Hamming weight. These two constraints are particularly applicable to SLC flash memory - the first constraint mitigates the problem of inter-cell interference, while the second constraint helps to alleviate the problems that arise from voltage drift of programmed cells. We give a construction for codes that satisfy both constraints, then analyze properties of best-case codes that can come from this construction.**

I. Introduction

Flash memory has become increasingly popular in recent years due to its fast read and write speeds and its low power consumption. The memory consists of a grid of floating-gate transistors called *cells*. To program a cell, a voltage is applied to the transistor in order to inject charge into the floating gate. The amount of charge stored in the cell determines the *cell level*. Thus, the level of any cell can be increased by injecting charge into that cell. Flash memory with cells that can distinguish between two levels can store one bit per cell; this type of memory is called single-level cell (SLC) flash (this is a misnomer, as the cell consists of one bit rather than one level). Cells that can store four levels, or two bits, are referred to as multi-level cells (MLC), and cells with eight levels are called triple-level cells (TLC). A larger number of bits per cell require a very precise level of charge injection, but can drastically increase the storage capacity of the memory.

For years, the feature size of these memories has been rapidly shrinking, resulting in an increase in storage capacity. In recent years, the very small feature size has presented a number of obstacles. One of the largest of these obstacles is *inter-cell interference* (ICI). This phenomenon occurs when a cell is programmed; when charge is injected into a cell, that cell's physical neighbors also receive a small increase in charge due to the parasitic capacitance between cells. Put another way, a *victim cell*'s charge will increase when its physical neighbors are programmed [3], [7].

For SLC memory, the worst-case pattern for three consecutive cells is the 101 pattern. Here, the victim cell is 0 (low) and its two neighbors are both 1 (high). The increase in charge from these neighboring cells may be enough to increase the victim cell from 0 to 1. For cells with q levels, the worst-case scenario is (q-1) 0 (q-1). Constrained codes that eliminate the worst-case patterns can be considered to reduce the impact of ICI [1]; these codes are called *ICI-free codes*.

A second obstacle to reliably storing data in flash memory is *voltage drift*. Over time, the charge that is stored in the cells leaks out of the floating gates. With enough time, this leakage can cause a cell's level to decrease, resulting in *retention errors*. As the feature size of flash memory shrinks, this problem becomes more pronounced - it takes less time for a large number of retention errors to appear.

Dynamic read thresholds have been proposed in [10] to remedy the errors caused by voltage drift. Under the assumption that all cells leak charge at the same rate, the relative levels of the cells remain unchanged. Both *constant-weight codes* [2] and *balanced codes* can thus be utilized to determine whether the read level should be adjusted; constant-weight codes must contain some constant proportion p of ones in every codeword, while balanced codes are a special case of constant-weight codes where there are an equal number zeros and ones in each codeword ($p = 1/2$). Thus, for a constant-weight code of length n and ratio p, if we read fewer than np ones in a codeword, we deduce that the cell levels have dropped, and we should read at a lower threshold.

While other types of errors requiring error correction can occur, we limit our error model to the two problems outlined above and leave error correction capability to future work.

The rest of the paper will be organized as follows: In Section II, we will present a construction for constant-weight ICI-free codes that is similar to the balancing of q-ary codewords (see [6] for more details on binary balancing and [9] for details on q-ary balancing). We then calculate the coding rate of this construction. In Section III, we examine the best-case codes that come from our construction; these are codes that are constructed to give high rates by using optimal parameters as the input to the construction.

II. Constrained Code Construction A

We wish to construct a codebook with a corresponding encoder and decoder that satisfies the following two constraints:

- For code length n and some fixed p, $0 \leq p \leq 1$, every codeword has constant Hamming weight pn.
- The subsequence 101 does not appear in any codeword.

There is no restriction on p in this constraint; in our code, it is automatically selected as a function of several parameters chosen by the user. These constraints are jointly known as the *constant-weight ICI-free constraint*.

The idea behind this encoder is to generate an alphabet of symbols, each of which represents a binary string that does not contain the substring 101. Words are then generated using this alphabet, and the resulting overall binary sequence is manipulated until the sequence has a fixed proportion p of ones by replacing symbols from the 101-free alphabet with other symbols from the same alphabet. The encoder and decoder are explained in more detail below.

Let $\mathcal{C}$ be a binary constrained code of length n, rate $R_{\mathcal{C}}$, and codebook size $|\mathcal{C}| = S$, with codewords $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{S-1}$ indexed such that $w(\mathbf{c}_0) \leq w(\mathbf{c}_1) \leq \ldots \leq w(\mathbf{c}_{S-1})$, where $w(\mathbf{c})$ is the Hamming weight of $\mathbf{c}$. The code satisfies the following constraints, henceforth collectively known as the *contiguous-weight 101-free constraint*:

1) No codeword contains the forbidden sequence 101.
2) No codeword ends with the subsequence 10.
3) No codeword begins with the subsequence 01.
4) For $0 \leq i \leq S-2$, either $w(\mathbf{c}_{i+1}) = w(\mathbf{c}_i) + 1$ or $w(\mathbf{c}_{i+1}) = w(\mathbf{c}_i)$. That is, there exists at least one codeword of every Hamming weight between the minimum-weight $w(\mathbf{c}_0)$ and the maximum-weight $w(\mathbf{c}_{S-1})$.

The second and third constraints together forbid 101 from appearing between any two concatenated codewords. The fourth constraint requires the set of Hamming weights for all codewords to be contiguous.

Define $p^*_{\mathcal{C}} = \frac{1}{nS}\sum_{\mathbf{c}\in\mathcal{C}} w(\mathbf{c})$; that is, $p^*_{\mathcal{C}}$ is the average proportion of ones over all codewords in $\mathcal{C}$.

Addition between a vector $\mathbf{c}_i \in \mathcal{C}$ and an integer j is defined by adding the integer to the index modulo S, i.e., $\mathbf{c}_i + j = \mathbf{c}_{i+j \bmod S}$. Thus, incrementing a codeword means choosing the next codeword in the ordered set of codewords.

We choose a second binary constrained code $\mathcal{C}'$ with length n' and with at least mS messages, where m is a freely chosen integer. The code should satisfy the following constraints:

1) Every codeword has a proportion of ones equal to q^* for $0 \leq q^* \leq 1$; that is, every codeword has Hamming weight $n'q^*$.
2) No codeword contains the forbidden sequence 101.
3) No codeword begins with the subsequence 01.

The code should be chosen to minimize n' for fixed m and code $\mathcal{C}$.

Using codes $\mathcal{C}$ and $\mathcal{C}'$, we now describe the encoder and decoder for the overall constrained code.

Encoder A

1) Use the code $\mathcal{C}$ to encode m messages, each containing $nR_{\mathcal{C}}$ bits of information. Call the result $\mathbf{x} = \mathbf{c}_{i_0}\mathbf{c}_{i_1}\ldots\mathbf{c}_{i_{m-1}}$, where each $\mathbf{c}_i$ is a codeword in $\mathcal{C}$.
2) Fix the Hamming weight of this sequence to $\lfloor mnp^*_{\mathcal{C}} \rfloor$ using the following steps.
   a) Initialize $j = 0$ and $k = 0$.
   b) Add $j+1$ to the first k symbols in $\mathbf{x}$ and add j to the remaining $m-k$ symbols and call the result $\mathbf{y}'$, i.e.,
   $$\mathbf{y}' = \mathbf{c}_{i_0+(j+1)}\mathbf{c}_{i_1+(j+1)}\cdots \mathbf{c}_{i_{k-1}+(j+1)}\mathbf{c}_{i_k+j}\mathbf{c}_{i_{k+1}+j}\ldots\mathbf{c}_{i_{m-1}+j}.$$
   c) If the sum of the Hamming weights of the symbols in $\mathbf{y}'$ is $\lfloor mnp^*_{\mathcal{C}} \rfloor$, we are finished; go to step 3.
   d) Update $\mathbf{x}$ by setting $\mathbf{x} = \mathbf{y}'$.
   e) Set $k = k + 1$
   f) If $k = m-1$, set $k = 0$ and $j = j+1$.
   g) Go to (b).
3) Using $\mathcal{C}'$, encode the message $jm + k$, with j and k obtained in step 2, to obtain sequence $\mathbf{r}$ of length n'.
4) Let $\mathbf{y} = \mathbf{y}'\mathbf{r}$. Then $\mathbf{y}$ has Hamming weight $\lfloor mnp^*_{\mathcal{C}} \rfloor + q^*$ and does not contain the subsequence 101.

The decoder is a straight-forward reversal of the encoder:

Decoder A

1) Split the codeword $\mathbf{y}$ into m binary sequences $\mathbf{c}_{i_0}, \mathbf{c}_{i_1}, \ldots, \mathbf{c}_{i_{m-1}}$ each of length n, and a length-n' binary sequence $\mathbf{r}$, so that $\mathbf{y} = \mathbf{c}_{i_0}\mathbf{c}_{i_1}\ldots\mathbf{c}_{i_{m-1}}\mathbf{r}$.
2) Decode $\mathbf{r}$ using the decoder of $\mathcal{C}'$, and convert the resulting binary vector to an integer t.
3) Find integers j and k, $0 \leq j \leq S-1$ and $0 \leq k \leq m-1$, such that $jm + k = t$.
4) Let
$$\mathbf{x} = \mathbf{c}_{i_0-(j+1)}\mathbf{c}_{i_1-(j+1)}\cdots \mathbf{c}_{i_{k-1}-(j+1)}\mathbf{c}_{i_k-j}\mathbf{c}_{i_{k+1}-j}\ldots\mathbf{c}_{i_{m-1}-j}.$$
5) Decode each symbol in $\mathbf{x}$ using the decoder of $\mathcal{C}$ to obtain the original m messages.

Theorem 1. *Using the preceding encoder and decoder pair, we can encode $nR_{\mathcal{C}}m$ bits into a constant-weight codeword of length $nm + n'$ and Hamming weight $\lfloor mnp^*_{\mathcal{C}} \rfloor + q^*$ such that the subsequence 101 does not appear. Here, n and $R_{\mathcal{C}}$ are the length and rate, respectively, of the code $\mathcal{C}$ defined above, n' is the length of the code $\mathcal{C}'$, $p^*_{\mathcal{C}}$ is the average proportion of ones over all codewords in the code $\mathcal{C}$, and q^* is the proportion of ones in every codeword in the code $\mathcal{C}'$.*

Proof. Consider the sequences generated by step 2 when $k = 0$. Let $\mathbf{v}_{j'}$ be the sequence when $j = j'$ and $k = 0$. Then $\mathbf{v}_{j'}$ is the sequence created by incrementing every symbol in $\mathbf{v}_0$ by j'. For a given symbol position, every symbol in $\mathcal{C}$ is used exactly once across all sequences $\mathbf{v}_{j'}$. Thus, the average Hamming weight among these sequences is

$$\frac{\sum_{j'=0}^{S-1} w(\mathbf{v}_{j'})}{S} = \frac{m\sum_{j'=0}^{S-1} w(\mathbf{c}_{j'})}{S} = mnp^*_{\mathcal{C}}, \qquad (1)$$

where the second equality comes from the fact that the average Hamming weight of all the codewords in $\mathcal{C}$ is $np^*_{\mathcal{C}}$.

Note that step 2 is cyclic; when $j = S$ and $k = 0$, we construct the same sequence as when $j = 0$ and $k = 0$. Then there must exist iterations a and b, $a < b$, such that iteration a results in a sequence with weight less than $\lfloor mnp^*_{\mathcal{C}} \rfloor$ and iteration b results in a sequence with weight greater than $\lfloor mnp^*_{\mathcal{C}} \rfloor$. Because the only allowable increase in Hamming weight during each iteration of step 2 is an increase of 1, we must reach a sequence with weight equal to $\lfloor mnp^*_{\mathcal{C}} \rfloor$ on some iteration between a and b.

The sequence $\mathbf{y}'$ constructed in step 2 clearly does not contain the subsequence 101; no codeword in $\mathcal{C}$ contains the subsequence 101, and the second and third constraints of

TABLE I
NUMBER OF WORDS OF EACH HAMMING WEIGHT IN $\mathcal{C}$ FOR $n = 12$.

Hamming Weight	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of Elements	1	10	39	84	120	126	105	64	45	10	11	0	1

the contiguous-weight 101-free constraint prevent 101 from appearing between codewords.

In step 3, j is at most $S-1$ and k is at most $m-1$, so $jm+k$ is an integer between 0 and $mS-1$. Thus, each message can be uniquely encoded into code $\mathcal{C}'$, which contains at least mS codewords.

Finally, in step 4, it can be easily verified that 101 does not appear when the two sequences $\mathbf{y}'$ and $\mathbf{r}$ are concatenated. The sequence $\mathbf{y}'$ has weight $\lfloor mnp^*_{\mathcal{C}} \rfloor$, and $\mathbf{r}$ has weight q^*, and so $\mathbf{y}'\mathbf{r}$ has weight $\lfloor mnp^*_{\mathcal{C}} \rfloor + q^*$. □

The overall rate of this code is $R = \frac{nR_{\mathcal{C}}m}{mn+n'} = \frac{m\log_2(S)}{mn+n'}$.

Remark 1. In the worst case, the encoder requires mS iterations. We can significantly improve on this worst-case; instead of incrementing only one of the m symbols on each iteration, we can increment all m symbols in a single iteration. Note that the resulting sequences are exactly those that are summed in Equation 1. When we reach an iteration where the average Hamming weight increases from less than $mnp^*_{\mathcal{C}}$ to greater than or equal to $mnp^*_{\mathcal{C}}$, we can start incrementing the symbols one at a time until we find this crossover point. The worst-case number of iterations is then $S+m$. However, because S is the number of messages in the code $\mathcal{C}$, we note that the encoding complexity is still quite high.

Example 1. There are 616 sequences of length 12 that do not contain the subsequence 101, do not start with the subsequence 01, and do not end with the subsequence 10. These sequences are broken down by Hamming weight in Table I. We eliminate the all-1 codeword from this list in order to satisfy the contiguous-weight requirement and choose the remaining 615 codewords as the code $\mathcal{C}$, with length $n = 12$ and rate $R_{\mathcal{C}} = \log_2(615)/12 \approx 0.7720$. The average proportion of ones in $\mathcal{C}$ is $p^*_{\mathcal{C}} \approx 0.4184$.

We start by selecting $m = 64$. Then the code $\mathcal{C}'$ must contain at least $mS = 39360$ codewords. The smallest code length that satisfies the constraints of $\mathcal{C}'$ and that contains at least 39360 codewords is $n' = 23$. This code $\mathcal{C}'$ does not have any codewords that contain 101 or begin with 01, and it contains 61078 codewords, all of weight $q^* = 9$. At this point, we can optionally increase m without increasing n' so long as $mS \leq 61078$. Thus, we choose $m = 99$ so that $mS = 60885$. Then our final parameters are $n = 12$, $m = 99$, $S = 615$, and $n' = 23$. Then the overall rate is $R = \frac{m\log_2(S)}{mn+n'} \approx 0.7574$.

Lemma 2. *As the number of concatenated sequences m approaches infinity, the overall rate a code from Construction A can achieve approaches $R_{\mathcal{C}}$. That is, $\lim_{m\to\infty} R = R_{\mathcal{C}}$.*

Proof. Suppose we naively construct the code $\mathcal{C}' = \{\mathbf{x}_{i_1}\mathbf{x}_{i_2}\ldots\mathbf{x}_{i_t} \mid i_j \in \{0,1\}, 1 \leq j \leq t\}$. Here, $\{\mathbf{x}_0, \mathbf{x}_1\} = \{1^{2a}0^{2b-2a}, 0^{2b-2a}1^{2a}\}$, $t = \lceil \log_2(mS) \rceil$, and a and b are integers chosen so that $q^* = \frac{a}{b}$. It is easy to verify that $\mathcal{C}'$ satisfies all the required constraints. The length of $\mathcal{C}'$ is $n' = 2b\lceil \log_2(mS) \rceil$. Using this $\mathcal{C}'$, our overall code rate is

$$R = \frac{m\log_2 S}{mn + 2b\lceil \log_2(mS) \rceil}. \quad (2)$$

Then as $m \to \infty$, $\lim_{m\to\infty} R = \frac{\log_2 S}{n} = R_{\mathcal{C}}$. □

TABLE II
CODEBOOK SIZE S AND OVERALL RATE R FOR A RANGE OF n AND m VALUES

n	m	S	n'	R
6	338	20	19	0.7136
7	289	36	20	0.7313
8	253	64	21	0.7423
9	225	113	22	0.7497
10	202	199	23	0.7551
11	184	350	24	0.7593
12	168	615	24	0.7630
13	155	1080	25	0.7656
14	144	1896	26	0.7679
15	134	3328	27	0.7697
16	126	5841	28	0.7713
17	118	10251	29	0.7726
18	112	17990	30	0.7738
19	106	31571	31	0.7747
20	100	55404	32	0.7755

III. ANALYSIS OF OPTIMAL CONSTRUCTIONS

We can choose for our code $\mathcal{C}$ any code that satisfies the contiguous-weight 101-free constraint. However, we now consider codes of length n with maximum rate (i.e., codes of length n with the maximum number of codewords). We derive the codebook size of such a code as a function of n, then show that when this maximum rate code $\mathcal{C}$ is used as the input to our construction, the rates of the codes produced approach the Shannon capacity [8] of the no-101 constraint as the block length approaches infinity. We also derive the average proportion of ones $p^*_{\mathcal{C}}$ of the maximum-rate code $\mathcal{C}$ as a function of n.

Theorem 3. *For code length $n \geq 2$, the highest-rate code $\mathcal{C}$ that satisfies the contiguous-weight 101-free constraint has codebook size*

$$S = \frac{2z_1^n + z_1^{n-2}}{(z_1 - z_2)(z_1 - z_3)} + \frac{2z_2^n + z_2^{n-2}}{(z_2 - z_1)(z_2 - z_3)} + \frac{2z_3^n + z_3^{n-2}}{(z_3 - z_1)(z_3 - z_2)} - 1, \quad (3)$$

where z_1, z_2, and z_3 are the roots of the cubic polynomial $z^3 - 2z^2 + z - 1$. Furthermore, the code of this size is unique.

TABLE III
THE FIRST SEVERAL VALUES OF $a_{n,00}$.

n	2	3	4	5	6	7	8	9	10
$a_{n,00}$	1	2	3	5	9	16	28	49	86

Proof. Let $\mathcal{S}_{n,\mathbf{v}}$ be the set of all binary vectors of length n that begin with the sequence $\mathbf{v}$, do not contain the subsequence 101, and do not end with 10. Note that the sequence $1^k 0^{n-k}$ satisfies conditions 1–3 of the contiguous-weight 101-free constraint for $0 \le k \le n-2$. Also note that every sequence of Hamming weight $n-1$ violates one of the conditions of the contiguous-weight 101-free constraint. Finally, the all-1's sequence of Hamming weight n satisfies conditions 1–3. Thus, $\mathcal{S}_n = \mathcal{S}_{n,00} \cup \mathcal{S}_{n,1} \setminus \{1^n\}$ is the largest codebook that satisfies the contiguous-weight 101-free constraint. If we let $a_{n,\mathbf{v}} = |\mathcal{S}_{n,\mathbf{v}}|$, then the size of this set can be expressed as

$$|\mathcal{S}_n| = a_{n,00} + a_{n,1} - 1. \tag{4}$$

It remains to derive expressions for $a_{n,00}$ and $a_{n,1}$.

It is easy to show that the sets $\mathcal{S}_{n,00}$, $\mathcal{S}_{n,01}$, and $\mathcal{S}_{n,1}$ can be recursively constructed for $n \ge 3$. Using the corresponding equations for the sizes of these sets, we can find a recurrence relation for $a_{n,00}$ with the solution

$$a_{n,00} = c_1 z_1^n + c_2 z_2^n + c_3 z_3^n, \tag{5}$$

where c_i are constants and z_i are the roots of the characteristic equation $z^3 - 2z^2 + z - 1 = 0$. The expressions for these roots can be written explicitly using the standard formula for cubic roots (see e.g. [4]). We can also find an expression for $a_{n,1}$ in terms of $a_{n,00}$ for $n \ge 5$.

$$a_{n,1} = a_{n,00} + a_{n-2,00} \tag{6}$$

Table III gives the values of $a_{n,00}$ for $2 \le n \le 10$.

Using the values given in Table III, we can work backwards with the recurrence relation and find initial conditions $a_{2,00} = 1$, $a_{1,00} = 0$, and $a_{0,00} = 0$. These initial conditions together with (5) gives the matrix equation $Z\mathbf{c} = \mathbf{a}$:

$$\begin{bmatrix} 1 & 1 & 1 \\ z_1 & z_2 & z_3 \\ z_1^2 & z_2^2 & z_3^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{7}$$

Because Z is a Vandermonde matrix, the inverse is easily found, and we can quickly solve for c_1, c_2, and c_3:

$$c_1 = \frac{1}{(z_1 - z_3)(z_1 - z_2)} \qquad c_2 = \frac{1}{(z_2 - z_1)(z_2 - z_3)}$$
$$c_3 = \frac{1}{(z_3 - z_1)(z_3 - z_2)}.$$

Combining this result with (4) yields (3). □

We can use this result to show that when maximum-rate codes are used as the input to our construction, the resulting codes will approach the capacity of the 101-free constraint as the block length grows. We first derive an expression for

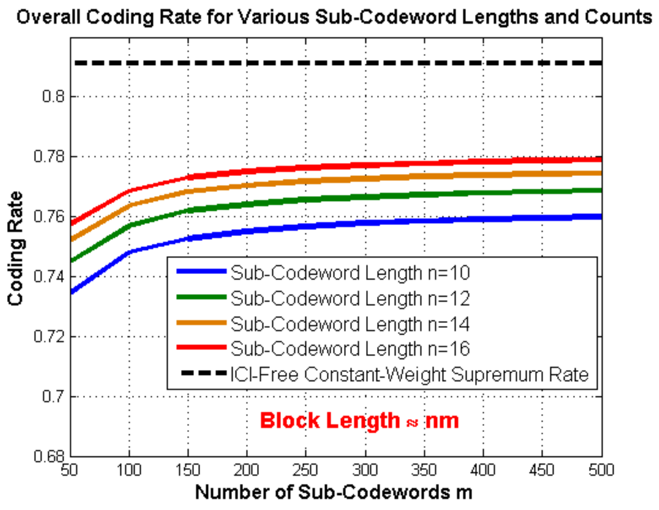

Fig. 1. Rates the construction achieves using optimal sub-codes. As n and m both approach infinity, the construction produces codes with rates that approach capacity.

the capacity of the 101-free constraint. We then show that the supremum of the rates of all constant-weight ICI-free codes is equal to the capacity of the 101-free constraint.

Lemma 4. *The capacity of the 101-free constraint is* Cap('no-101') $= \log_2(z_1)$, *where* z_1 *is the unique real solution to the polynomial equation* $z^3 - 2z^2 + z - 1 = 0$.

Proof. The proof is omitted due to space constraints, but the result can be verified by finding the log of the Perron-Frobenius eigenvalue of the adjacency matrix for this constraint. See [5] for the details on finding the capacity of a constrained system. □

Theorem 5. *The supremum of the rates of all possible constant-weight ICI-free codes is* $R_{\text{sup}} = \log_2(z_1)$, *where* z_1 *is the unique real solution to the polynomial equation* $z^3 - 2z^2 + z - 1 = 0$. *Furthermore, the rate of codes from Construction A approaches this rate.*

Proof. We know from Lemma 2 that the rates of codes from our construction approach $\frac{\log_2 S}{n}$ as $m \to \infty$. Then we just need to show $\lim_{n\to\infty} \frac{\log_2 S}{n} = \log_2(z_1)$. We can see from the formula for cubic roots [4] that $z^3 - 2z^2 + z - 1 = 0$ has one real solution z_1 and two complex solutions z_2 and z_3. We note that $|z_2| = |z_3| < 1$. Combining this fact with (3) gives

$$\lim_{n\to\infty} \frac{\log_2 S}{n} = \log_2(z_1). \tag{8}$$

□

Figure 1 shows the coding rate the construction achieves for several values of sub-code length n and number of sub-codewords m using optimal sub-codes. As n and m approach infinity, code rates of this construction approach capacity.

Theorem 6. *The average proportion of ones over all codewords in the maximum-rate code that satisfies the contiguous-*

weight 101-free constraint is

$$p^*_{\mathcal{C}} = \frac{\sum_{k=0}^{n-2} k d_{n,k}}{n \sum_{k=0}^{n-2} d_{n,k}} \tag{9}$$

where

$$d_{n,k} = \sum_{i=0}^{\min(2k+2, n-k)} (-1)^i \binom{n-i}{k} \sum_{j=0}^{\lfloor i/2 \rfloor} \binom{k+1}{i-j} \binom{i-j}{j}$$

Proof. Let $Q_{n,\mathbf{v}}(x) = \sum_{k=0}^{\infty} d_{n,k,\mathbf{v}} x^k$ be the generating function where $d_{n,k,\mathbf{v}}$ is the number of binary sequences of length n and Hamming weight k that begin with $\mathbf{v}$ and satisfy the following two properties:

1) 101 does not appear in the sequence.
2) The sequence does not end with 10.

Let $\mathcal{S}_{n,k,\mathbf{v}}$ be the set of all binary vectors of length n and Hamming weight k that begin with the sequence $\mathbf{v}$ and satisfy the two constraints above, and note that $|\mathcal{S}_{n,k,\mathbf{v}}| = d_{n,k,\mathbf{v}}$. Let $d_{n,k}$ be the number of sequences that satisfy the two constraints above and do not begin with 01, i.e., $d_{n,k} = d_{n,k,00} + d_{n,k,1}$. Following similar steps to those found in the proof of Theorem 3, we derive the recurrence relation for $n \geq 5$

$$d_{n,k} = d_{n-1,k} + d_{n-1,k-1} - d_{n-2,k-1} + d_{n-3,k-1}. \tag{10}$$

Let

$$Q_n(x) = \sum_{k=0}^{\infty} d_{n,k} x^k \tag{11}$$

be the generating function of $d_{n,k}$ for all k. Plugging (10) into this equation yields a new recurrence relation (after some minor manipulation)

$$Q_n(x) = (x+1)Q_{n-1}(x) - xQ_{n-2}(x) + xQ_{n-3}(x). \tag{12}$$

Let $A(x,y)$ be the generating function of $Q_n(x)$, i.e.,

$$A(x,y) = \sum_{n=0}^{\infty} Q_n(x) y^n. \tag{13}$$

If we first find explicit expressions for $Q_0(x)$, $Q_1(x)$, and $Q_2(x)$ and plug our new recurrence relation (12) into (13), we can solve for $A(x,y)$ as

$$A(x,y) = \sum_{k=0}^{\infty} \frac{y^k (1-y+y^2)^{k+1}}{(1-y)^{k+1}} x^k. \tag{14}$$

Let $R_k(y)$ be the coefficient of x^k. Our goal is to express $R_k(y)$ as a power series in y and plug the result into (14). This will yield a two-dimensional generating function in x and y, where the coefficient of $x^k y^n$ is $d_{n,k}$, the number of sequences of length n and Hamming weight k that satisfy the first three conditions of the contiguous-weight 101-free constraint.

By applying the identity $\frac{1}{(1-y)^{k+1}} = \sum_{m=0}^{\infty} \binom{k+m}{m} y^m$ and with repeated use of the binomial formula and a series of change of variable, we express $R_k(y)$ as a power series in y:

$$R_k(y) = \sum_{n=k}^{\infty} \left[\sum_{i'=0}^{\min(n-k, 2k+2)} \binom{n-i'}{n-i'-k} (-1)^{i'} \cdot \sum_{j=0}^{\lfloor i'/2 \rfloor} \binom{k+1}{i'-j} \binom{i'-j}{j} \right] y^n \tag{15}$$

This is a generating function where the coefficient in front of each y^n is $d_{n,k}$. Thus, by recognizing that $\binom{n-i'}{n-i'-k} = \binom{n-i'}{k}$, we obtain (6).

Recall from the proof of Theorem 3 that the largest code $\mathcal{C}$ of code length n that satisfies the contiguous-weight 101-free constraint contains all length-n sequences that satisfy the first three conditions of the constraint except for the all-1 sequence. Also recall that there are no sequences of Hamming weight $n-1$ that satisfy the first three conditions. Hence, the total number of 1's over all sequences in $\mathcal{C}$ can be computed by counting the number of sequences of Hamming weight k for $0 \leq k \leq n-2$, and so the average proportion $p^*_{\mathcal{C}}$ of ones of all codewords in $\mathcal{C}$ is as in the statement of the theorem. □

IV. Conclusion

We considered two constraints designed to improve the error performance of flash memory: forbidding the 101 subsequence to eliminate inter-cell interference, and requiring all codewords to have the same Hamming weight so that voltage drift can be detected. We introduced a code construction that implements both constraints simultaneously and derived an expression for the coding rate our construction achieves. We then showed that the rates of codes from our construction will approach the capacity of the 'no-101' constraint when the optimal input parameters are chosen and the block length grows.

References

[1] A. Berman and Y. Birk, "Constrained flash memory programming," in *Proc. IEEE Int. Symp. Inform. Theory*, St. Petersburg, Russia, July - August 2011, pp. 2128 – 2132.

[2] A. Brouwer, J. Shearer, N. Sloane, and W. Smith, "A new table of constant weight codes," *IEEE Trans. Inform. Theory*, vol. 36, no. 6, pp. 1334–1380, November 1990.

[3] G. Dong, S. Li, and T. Zhang, "Using data postcompensation and predistortion to tolerate cell-to-cell interference in MLC NAND flash memory," *IEEE Trans. Circuits Syst.*, vol. 57, no. 10, pp. 2718 – 2728, October 2010.

[4] D. S. Dummit and R. M. Foote, *Abstract Algebra.* John Wiley & Sons, 2004.

[5] K. A. S. Immink, *Codes for Mass Data Storage Systems.* Shannon Foundation Publishers, The Netherlands, 2004.

[6] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inform. Theory*, vol. 32, no. 1, pp. 51 – 53, January 1986.

[7] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Lett.*, vol. 23, no. 5, pp. 264 – 266, May 2002.

[8] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379 – 423, 1948.

[9] T. G. Swart and J. H. Weber, "Efficient balancing of q-ary sequences with parallel decoding," in *Proc. IEEE Int. Symp. Inform. Theory*, Seoul, Korea, June–July 2009, pp. 1564–1568.

[10] H. Zhou, A. Jiang, and J. Bruck, "Error-correcting schemes with dynamic thresholds in nonvolatile memories," in *Proc. IEEE Int. Symp. Inform. Theory*, St. Petersburg, Russia, July-August 2011, pp. 2143 – 2147.