IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 13, 2021, accepted February 28, 2021, date of publication March 10, 2021, date of current version March 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065067

Synchronization of Variable-Length Constrained Sequence Codes

CONGZHE CAO AND IVAN FAIR, (Member, IEEE)
Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada

Corresponding author: Congzhe Cao (congzhe@ualberta.ca)

This work was supported by the Natural Science and Engineering Research Council (NSERC) of Canada and Alberta Innovates Technology Futures (AITF).

ABSTRACT We study the ability of recently developed variable-length constrained sequence codes to determine codeword boundaries in the received sequence upon initial receipt of the sequence and if errors in the received sequence cause synchronization to be lost. We first investigate construction of these codes based on the finite state machine description of a given constraint, and develop new construction criteria to achieve high synchronization probabilities. Given these criteria, we propose a guided partial extension algorithm to construct variable-length constrained sequence codes with high synchronization probabilities. With this algorithm we construct new codes and determine the number of codewords and coded bits that are needed to recover synchronization once synchronization is lost. We consider a large variety of constraints including the runlength limited (RLL) constraint, the DC-free constraint, the Pearson constraint and constraints for inter-cell interference mitigation in flash memories. Simulation results show that the codes we construct exhibit excellent synchronization properties, often resynchronizing within a few bits.

INDEX TERMS Constrained sequence codes, variable-length codes, capacity-approaching codes, construction, synchronization, error propagation.

I. INTRODUCTION

Constrained sequence (CS) codes have been widely used to increase the efficiency and reliability of data storage and digital communication systems such as optical recording, magnetic recording, flash memories, DNA-based storage, cable transmission, visible light communications and wireless energy harvesting, among other applications [1]–[4]. Since Shannon's 1948 paper [5], the design of CS codes has been an active research area where efficient CS codes that satisfy a great variety of constraints have been proposed. Although most CS codes in the literature are fixed-length codes [1]–[4], [6]–[16], recent advances show that variable-length CS codes have the potential to achieve higher code rates with simpler codebooks [17]–[26]. Since CS codes typically do not have strong error-correction capabilities, decoding of CS codes may result in error propagation. In practical systems CS codes are commonly used in conjunction with an interleaver and an error control code (ECC) such as an LDPC code [27], a polar code [28]–[31] or a product code [32]–[35] to overcome error propagation that may occur during CS decoding. Alternatively, automatic repeat request (ARQ) could be used such that when errors are detected at the output of the CS decoder, the system requests retransmission until the decoded sequence is error free.

Apart from their advantages, variable-length CS codes have the drawback that because of noise that occurs during transmission, erroneously received sequences may result in loss of codeword-boundary synchronization at the decoder. Mis-synchronization typically results in insertion or deletion errors, which may cause burst errors at the output of the CS decoder that may be difficult for the ECC to correct. To enable practical implementation of variable-length CS codes, we aim to develop codes with good synchronization properties, to make it feasible for ECCs to correct errors that occur at the output of the CS decoder. Alternatively with ARQ, it is desired that error propagation be limited to within the current received packet, to ensure that subsequent packets are unaffected.

In this paper, we consider the variable-length CS codes constructed by the approaches in [20]–[25] where the design guideline in those papers has been to achieve the highest

The associate editor coordinating the review of this manuscript and approving it for publication was Rui Wang.

possible code rate. We show that different strategies should be considered when we aim to develop codes that achieve both high efficiency and good synchronization properties. We consider a variety of constraints, including the runlength limited (RLL) constraint, constraints that mitigate ICI in flash memories, the Pearson constraint, and the DC-free constraint. The contributions of this paper are summarized as follows.

- We show that states selected based on the criteria in [21]–[25] may not result in the best synchronization properties, and we develop criteria for selecting the specified state that results in superior synchronization.
- We develop an algorithm to perform partial extensions that result in a set of codewords with high code efficiency along with high synchronization probability.
- We evaluate the synchronization properties of the proposed codes, including the number of codewords and number of coded bits that the receiver typically requires in order to regain synchronization once it is lost.
- We show that our proposed codes have good synchronization properties because they regain synchronization quickly in order to reduce the required error control capabilities of the outer ECCs.

We begin with necessary background information.

II. PRELIMINARIES

A. CS CODING THEORY

RLL and DC-free codes are two widely used classes of CS codes. RLL coded sequences have the property that the number of bits between transitions is bounded. They can be constructed by first generating a (d, k) sequence, where d and k denote the minimum and maximum number of logic zeros between consecutive logic ones, followed by differential encoding that encodes a logic one as a change in value and a logic zero as no change [1]. DC-free codes are designed so that the spectral components at low frequency are suppressed to match the characteristics of the physical channel. In the time domain, the running digital sum (RDS) of the DC-free encoded sequence is limited to N different values [1]. RDS is the ongoing summation of encoded bit weights in the sequence, where a logic one has weight $+1$ and a logic zero has weight -1. Other constraints include the Pearson constraint that is immune to unknown channel gain and offset [13]–[15], and constraints that mitigate inter-cell interference in flash memories [23], [36]–[39].

It is well known that a constraint can be described with a finite state machine (FSM) that contains states, edges and labels. For an FSM with $\mathcal{S}$ states, the directed graph underlying the constraint is described by an $\mathcal{S} \times \mathcal{S}$ adjacency matrix $D = \{d_{ij}\}$, where d_{ij} is the number of edges transitioning from state i to state j. The transition probability matrix is denoted by an $\mathcal{S} \times \mathcal{S}$ matrix $Q = \{q_{ij}\}$, where q_{ij} is the probability of transitioning from state i to state j. Based on D, the maxentropic transition probabilities and steady-state distribution can be evaluated to describe the statistical properties when the maximum amount of information is represented by the FSM [1], [40]. Based on the maxentropic transition probabilities and steady-state distribution, it is possible to obtain the maxentropic probability of each codeword, which is the occurrence probability of the codeword in the coded sequence when the maximum amount of information is conferred.

As outlined in [1], maxentropic transition probabilities in a FSM are given by

$$q_{ij} = \lambda_{\max}^{-1} d_{ij} \frac{p_j}{p_i} \tag{1}$$

where $1 \leq i, j \leq \mathcal{S}$, and p is the eigenvector of D associated with the eigenvalue λ_{max} which is the largest real root of the determinant equation

$$det[D - zI] = 0 \tag{2}$$

where I is an identity matrix and z represents roots of the determinant equation [1].

The maximum amount of information that can be carried in a sequence that satisfies the constraint is the *capacity* of the constraint $\mathfrak{C}$, which is defined as [5]

$$\mathfrak{C} = \lim_{m \to \infty} \frac{\log_2 N(m)}{m} \tag{3}$$

where $N(m)$ is the number of constraint-satisfying sequences of length m. Given the FSM description of the constraint, the capacity can be evaluated as $\mathfrak{C} = \log_2 \lambda_{max}$.

We denote $\widetilde{\mathfrak{C}}_M$ as the *maximum possible code rate* of a constrained coded sequence constructed with variable-length codewords with lengths from the set M. L_M denotes the set of word lengths in M, and $\widetilde{\mathfrak{C}}_M$ is determined as

$$\widetilde{\mathfrak{C}}_M = \log_2 \widetilde{\lambda}_{max} \tag{4}$$

where $\widetilde{\lambda}_{max}$ is the largest real root of the characteristic equation $\sum_{l_i \in L_M} \lambda^{-l_i} = 1$ and l_i is the length of the i-th word in M [1], [20]–[22]. We denote the maximum length of words in the minimal set as l_{max}, and $|M|$ as the number of words in M. Note that a word length l_i could appear more than once in L_M since there may be multiple words with the same length. The corresponding *maximum possible efficiency* is $\widetilde{\eta} = \widetilde{\mathfrak{C}}_M / \mathfrak{C}$.

B. CAPACITY-APPROACHING VARIABLE-LENGTH CS CODES

1) MINIMAL SETS AND EXTENSIONS

We now briefly review the construction of single-state capacity-approaching variable-length constrained sequence codes introduced in [20]–[24]. As discussed in [20]–[22], a critical step in construction of these codes is the formation of a minimal set from which words can be concatenated to generate constraint-satisfying codewords. A minimal set M_i can be established by enumerating all words exiting and re-entering a specific state i in the FSM. Codes constructed based on different states in the FSM have different maximum possible code rates, therefore as outlined in [21], [22] one needs to adhere to certain criteria to select the specified state

in the minimal set which has the highest maximum possible code rate.

Let M_i be a minimal set that contains K words. A *partial extension* of M_i, denoted $M_{i,p}$, is formed by concatenating all K words in M_i to any single word in M_i, generating $2K-1$ words. Subsequent partial extensions can be performed by appending the K words from M_i onto any word from the previous extension. We denote word $\mathcal{U} = \mathcal{W} + \mathcal{V}$ in the updated partial extension set as the result of concatenating word $\mathcal{V}$ in M_i to word $\mathcal{W}$ in the current set, where $+$ denotes concatenation. After performing partial extensions, we obtain a set of codewords that are instantaneously decodable due to the fact that no codeword in the minimal set M_i is a prefix of another, which also holds for any of its partial extensions $M_{i,p}$ [21]–[25].

2) NORMALIZED GEOMETRIC HUFFMAN (NGH) CODING

NGH coding [41]–[43] is used to assign codewords in $M_{i,p}$ to corresponding source words such that the maximum information density is approached. Starting with the desired codeword probabilities as the input probabilities, NGH coding merges the two smallest probabilities q_i and q_j according to the following rule to obtain the merged probability:

$$q_{merged} = \begin{cases} 2\sqrt{q_i q_j} & \text{if } q_i < 4q_j \\ q_i & \text{if } q_i \geq 4q_j. \end{cases} \tag{5}$$

The smaller probability is pruned from the Huffman tree when the lower condition is satisfied. As in the well-known Huffman construction technique, this process is repeated until a single value remains, and source words are assigned based on the merging pattern.

Given a one-to-one correspondence between variable-length source words and variable-length codewords, and assuming independent and equiprobable logic values in the source stream, the average code rate $\overline{R}$ is

$$\overline{R} = \frac{\sum_{s_i} 2^{-s_i} s_i}{\sum_{o_i} 2^{-s_i} o_i} \tag{6}$$

where s_i is the length of i-th source word that is mapped to the i-th codeword of length o_i. The efficiency of a variable-length code is defined as $\eta = \overline{R}/\mathfrak{C}$. After obtaining $\overline{R}$, NGH coding repeats the above process with updated input probabilities and with $\mathfrak{C}$ replaced by $\overline{R}$ when calculating the maxentropic probabilities in (1) – (2), until $\bar{R}$ converges.

Since different partial extensions result in different codebooks with different η, we establish parameters such as the maximum number of source words in the codebook n_{max}, or maximum codeword length l_{max}, and exhaustively search over all codebooks that satisfy these limits to find the one with the highest η.

Example 1: ($(d = 1, k = 3)$ RLL code [25]): the FSM of the $(d = 1, k = 3)$ RLL constraint is shown in Fig. 1. According to [21], [22], we choose state 1 *as the specified state upon which to construct the code; its minimal set is established as*

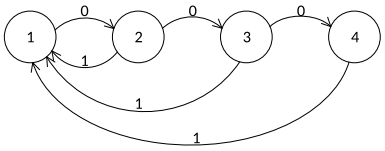

FIGURE 1. FSM of a $(d = 1, k = 3)$ RLL code.

$M_1 = \{01, 001, 0001\}$. *We may choose to directly perform NGH coding over the minimal set to construct the simple codebook shown in Table 1 which has efficiency* $\eta = 98.9\%$. *By performing extensions as illustrated in Fig. 2, we construct the code shown in Table 2 with* $\eta = 99.25\%$. *Note that the encoding and decoding processes are instantaneous with the words in this table. To compare, note that a widely used (1,3) RLL code is the MFM (Modified Frequency Modulation) code with* $\eta = 91\%$ *[1].*

TABLE 1. Codebook of a $(d = 1, k = 3)$ RLL code with efficiency of 98.9%.

Source word	Codeword
0	01
10	001
11	0001

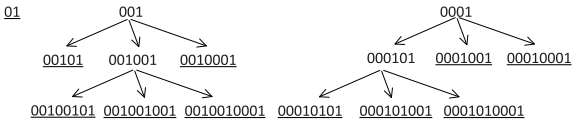

FIGURE 2. Partial extensions over M_1. Underlined words are codewords in the final code listed in Table 2.

TABLE 2. Codebook of a $(d = 1, k = 3)$ RLL code with efficiency of 99.25%.

Source word	Codeword	Source word	Codeword
0	01	11100	00010101
100	00101	11101	001001001
1010	0010001	11110	000101001
1011	0001001	111110	0010010001
1100	00010001	111111	0001010001
1101	00100101		

C. SYNCHRONIZATION

We wish to develop variable-length CS codes with good codeword synchronization properties, in which their codebooks have as many *synchronizing codewords* as possible. A synchronizing codeword guarantees that whenever it is received, the decoder achieves synchronization because it is guaranteed to correctly identify the end of this codeword and therefore maintain or recover synchronization, regardless of the correctness of the previously received codewords [44]. According to [44], a synchronizing codeword $C = c_1 c_2 \ldots c_n$ in the set of codewords from codebook C must satisfy the following two conditions:

Condition 1: $\forall X = x_1 x_2 \ldots x_m$ in C such that $m > n$ and C is a substring of X, $c_1 c_2 \ldots c_n = x_{m-n+1} x_{m-n+2} \ldots x_m$ and $c_1 c_2 \ldots c_n \neq x_i x_{i+1} \ldots x_{i+n-1}$ $\forall i \neq m - n + 1$.

Condition 2: $\forall j < n$ such that $c_1 c_2 \ldots c_j$ is a suffix of any codeword in C, $c_{j+1} c_{j+2} \ldots c_n$ is a valid codeword in C.

In particular, Condition 1 indicates that if a synchronizing codeword C is an internal bit string of any codeword X, the last bit of C must also be the last bit of X. This guarantees that correct identification of C results in correct recognition of the end of a codeword, hence synchronization is maintained or recovered. Condition 2 ensures that whenever mis-synchronization occurs that results in the receiver incorrectly identifying c_j as the end of a codeword, resynchronization is guaranteed to occur at the end of codeword C because c_n will be identified as the end of C. It is shown in [44] that some Huffman codes exhibit good synchronizing properties because their codewords re-synchronize the decoder regardless of the previous synchronization status.

We denote the synchronization probability (denoted as *sync probability*) P *of a codebook* C as the sum of occurrence probabilities of all synchronizing codewords. Note that synchronizing codewords appear more frequently in coded sequences constructed from codebooks with higher values of P, and therefore that decoders will typically re-synchronize more quickly once synchronization is lost, resulting in fewer burst errors for the outer ECC to correct. Our goal is therefore to construct codes with high synchronization probability.

To construct these codes, we first consider the minimal set, and then consider partial extensions of this minimal set to achieve high efficiency and high sync probability. Note that, prior to NGH coding, a minimal set and its partial extensions are not codebooks because their words have not been assigned source words, and therefore it is not possible to obtain their occurrence probabilities. However, because we expect good codes to have probabilities that are close to maxentropic, in our construction approach we approximate the sync probability P of a minimal set M and its partial extensions M_p as the sum of the maxentropic probabilities of each synchronizing word.

Lastly, we note that existence of synchronizing codewords is a sufficient but not necessary condition for the receiver to regain synchronization, since in some situations the receiver may correctly regain synchronization on nonsynchronizing words, as we will show in our simulation results. Thus, we expect there to be a strong correlation, but not necessarily a direct relationship, between the sync probability and synchronization performance.

III. MINIMAL SET CONSTRUCTION

Three criteria are introduced in [21], [22] to select the specified state that results in the minimal set with the highest possible code rate. In situations where we wish to construct codes with good synchronization properties, criteria should be developed that result in codebooks with high sync probabilities. In this section we consider selection of the specified state from the minimal set, and illustrate our selection criteria with a variety of constraints. In the next section we consider partial extensions of this minimal set.

Observation 1: In the FSM description of a constraint, if a state has a loop associated with itself, then selection of this state as the specified state is likely to correspond to a minimal set with a small sync probability, unless the loop corresponds to a synchronizing word.

Observation 1 arises when a single bit 1 or 0 that is associated with the specified state appears as a word in the minimal set. The word 1 or 0 is likely to violate Condition 1, therefore this case should be avoided unless codeword 0 or 1 is a synchronizing word. We demonstrate Observation 1 with the following examples.

Example 2: ((d, ∞) constraint): the FSM of the $(d, k = \infty)$ RLL constraint is shown in Fig. 3. We consider the $(d = 2, k = \infty)$ RLL constraint as an example. To maximize the efficiency of the code, in accordance with the criteria in [21], [22] we would select state 1 *as the specified state, and find the minimal set to be* $M_1 = \{0, 100\}$. *More generally, as is evident from Fig. 3, the minimal set of a* (d, ∞) *constraint is* $M_1 = \{0, 1\underbrace{0 \ldots\ldots\ldots\ldots\ldots\ldots}_{d \text{ zeros}}0\}$. *It can be verified that the second word is a synchronizing word. However, it can be observed that word* 0 *in* M_1 *is not a synchronizing word since it does not satisfy Condition 1. But it can also be verified that, if we select any other state as the specified state, the minimal set will have a sync probability of* 100%. *For example in the FSM of the* $(d = 2, k = \infty)$ *RLL constraint,* $M_2 = \{001, 0001, 00001, 000001, \ldots\}$ *in which every word in* M_2 *is a synchronizing word. However this minimal set contains an unlimited number of words, and therefore will result in a less efficient code than that constructed using* M_1 *[21], [22]. This demonstrates that a code optimized for efficiency may not be optimized for sync probability, and vice versa.*

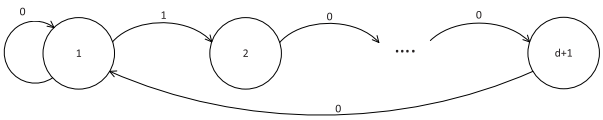

FIGURE 3. FSM of the $(d, k = \infty)$ RLL constraint.

Example 3: ($(0, k)$ constraint): the FSM of the $(d = 0, k)$ *RLL constraint is shown in Fig. 4. We consider the* $(d = 0, k = 3)$ *RLL constraint as an example. To maximize the efficiency of the code, in accordance with the criteria in [21], [22], we select state* 1 *as the specified state, and establish the minimal set as* $M_1 = \{1, 01, 001, 0001\}$. *It can be observed that for the* $(d = 0, k = 3)$ *RLL constraint, and*

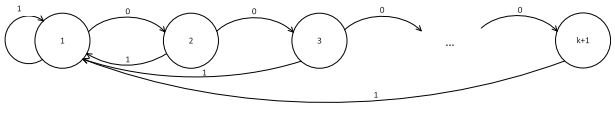

FIGURE 4. FSM of the $(d = 0, k)$ RLL constraint.

more generally for any $(d = 0, k)$ *constraint, every word in* M_1 *is a synchronizing word, and hence the sync probability is* 100%. *In this case, the selection criteria in [21], [22] also result in the minimal set with the highest sync probability. Note that this is in agreement with Observation* 1 *since the single-bit word* 1 *associated with state* 1 *is a synchronizing word.*

Observation 2: In the FSM description of a constraint, if outgoing edges and incoming edges of a state correspond to the same bit sequence, it is likely that selection of this state as the specified state results in a minimal set with few synchronizing words. This occurs because the prefix of a codeword $\mathcal{W}$ will be suffix of one or more codewords in the minimal set, and the remaining bits of $\mathcal{W}$ may not be a valid codeword, thus violating Condition 2.

Observation 2 arises when a word in the minimal set (that corresponds to the outgoing edges) can be divided into a suffix of another word (that corresponds to the incoming edges) plus the remaining bits, and therefore it might occur that these remaining bits are not a valid word, which violates Condition 2. We explain Observation 2 with the following example, in which we use the notation $\mathcal{W}\backslash\zeta$ to represent a substring of codeword $\mathcal{W}$ where the prefix ζ of $\mathcal{W}$ is excluded, and denote $M\backslash\mathcal{W}$ as the set of words in the minimal set M where word $\mathcal{W}$ is excluded.

Example 4: (general (d, k) *constraints): we consider general* (d, k) *constraints where* $d \neq 0$ *and* $k \neq \infty$. *The corresponding FSM is shown in Fig. 5. According to the criteria in [21], [22], any state from states* 1 *to* $d + 1$ *can be selected as the specified state because they all result in the highest maximum possible code rate. For example, the minimal set with state* 1 *as the specified state is* $M_1 = \{\underbrace{00\ldots0}_{d\ zeros}1, \underbrace{00\ldots0}_{d+1\ zeros}1, \underbrace{00\ldots0}_{d+2\ zeros}1, \ldots, \underbrace{00\ldots0}_{k\ zeros}1\}$. *Since every word is a synchronizing word, the sync probability is* 100%. *However, different from the criteria in [21], [22], as introduced in Observation 2, state* $\sigma, 1 < \sigma \leq d$ *should not be selected as the specified state. Consider state* 2 *as an example. One of the words* $\mathcal{W}$ *in its minimal set is* $\underbrace{00\ldots0}_{d-1\ zeros}10$ *which is not a synchronizing word, since* 0 *is a suffix of* $\mathcal{W}$ *itself but the minimal set does not contain the word* $\mathcal{W}\backslash 0 = \underbrace{00\ldots0}_{d-2\ zeros}10$. *Therefore, Condition* 2 *is not satisfied and hence* $\mathcal{W}$ *is not a synchronizing word, making the sync probability of* M_2 *less than* 100%. *A similar observation can be made for states* $\sigma, 2 \leq \sigma \leq d$, *where we can see that these states have both an outgoing edge and an incoming edge that are associated with bit zero, which violates Condition 2.* $\mathcal{W}$ *could be a synchronizing word iff* $\mathcal{W}\backslash 0$ *was a valid word, which is not the case.*

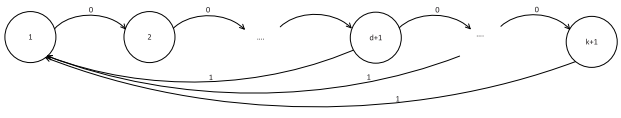

FIGURE 5. FSM of general (d, k) RLL constraints, $d \neq 0, k \neq \infty$.

Based on the above observations and examples, we propose the following criteria to select the specified state that results in high sync probability.

Criterion 1: if a state has a loop associated with itself, this state should not be selected as the specified state unless the loop corresponds to a synchronizing word.

Criterion 2: if outgoing edges and incoming edges of a state correspond to the same bit sequence, this state should not be selected as the specified state.

Note that these two criteria are guidelines for general constraints when selecting the specified state. Therefore one should consider the specific characteristics of each constraint when selecting the specified state. In addition to the (d, k) RLL constraints that we have discussed, we now illustrate state selection for a variety of other constraints.

A. CONSTRAINTS THAT MITIGATE ICI IN FLASH MEMORIES

In flash memories, one of the dominant errors is due to inter-cell interference (ICI) [23], [36]–[39], because variations of the electrical charge on one floating-gate transistor impacts the voltages of its neighboring transistors via the parasitic capacitance-coupling effect. Many constraints have been exploited in ICI mitigation [23], [36]–[39]. CS codes that forbid the pattern 101 have been designed to limit ICI [23], [36]–[39]; the FSM describing this constraint is shown in Fig. 6.

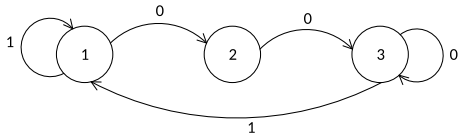

FIGURE 6. FSM of the constraint that forbids pattern 101 for ICI mitigation.

According to the criteria in [21], [22], states 1 and 3 are equally preferred as the specified state due to the fact that they both result in the highest maximum possible code rate. However, according to Criterion 1, state 3 is inferior to state 1 in terms of sync probability since state 3 has a loop associated with itself that corresponds to word 0, and the word 0 is not a synchronizing word. A closer look at $M_3 = \{0, 100, 1100, 11100, \ldots\}$ reveals that word 0 is not a synchronizing word, since 0 does not satisfy Condition 1. However, every word in $M_1 = \{1, 001, 0001, 00001, \ldots\}$ is a synchronizing word, including the one-bit word on the self-loop on state 1, resulting in the sync probability of M_1 to be 100%. Therefore, considering the criteria in both [21], [22] and in this paper, state 1 is preferred.

Now consider ICI mitigation in multi-level cell (MLC) flash memories, where each coded symbol can be 0, 1, 2 or 3 and the pattern 303 is forbidden by the constraint. The FSM that represents this constraint is shown in Fig. 7.

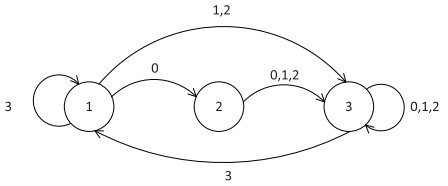

FIGURE 7. FSM of the constraint that mitigates ICI in MLC flash memories.

Different from the FSM in Fig. 6, according to the study in [21], [22], state 3 is better than state 1 in terms of maximum possible code rate. As demonstrated in the previous example, however, based on Criterion 1, M_3 has the word 0 that is not a synchronizing word whereas as we demonstrate below, every word in M_1 is a synchronizing word. This results in the sync probability of M_1 to be 100%. Therefore, we note that in general there is a tradeoff between the maximum possible code rate and the sync probability, and it is the choice of the system designer as to which to optimize when selecting the specified state and constructing the corresponding minimal set.

To show that the sync probability of M_1 is 100%, consider first the quaternary bit 3 in M_1. It can be observed that 3 does not violate the synchronization conditions and hence is a synchronizing word. Consider the set $M_1\backslash 3$. All words in M_1 end with a quaternary bit 3, and no word in $M_1\backslash 3$ starts with a quaternary bit 3, therefore Condition 2 is satisfied for all words in $M_1\backslash 3$. Furthermore, since the quaternary bit 3 only appears at the end of each word, Condition 1 is satisfied for all words in $M_1\backslash 3$. Thus, M_1 has a sync probability of 100%.

B. THE PEARSON CONSTRAINT

The Pearson constraint that is immune to unknown channel gain and offset can be regarded as a type of T-constrained code where each of the T pre-defined symbols appears at least once in every codeword [45]. As discussed in [14], [18], [24], a known construction for q-ary Pearson codes is to ensure that every q-ary codeword has at least one symbol "0" and one symbol "1".

The FSM of the binary Pearson constraint is shown in Fig. 8 [24]. According to the criteria in [21], [22], we select state N as the specified state to achieve the highest maximum possible code rate. However, it can be observed that state N does not satisfy Condition 2 since both an outgoing edge and an incoming edge correspond to bit 0 (and bit 1). Therefore, state N results in a sync probability lower than 100%. The minimal set is $M_N = \{01, 10, 001, 110, 0001, 1110, \ldots\}$. It can be verified that every word in M_N other than the words 01 and 10 is a synchronizing word, and the sync probability is 50%.

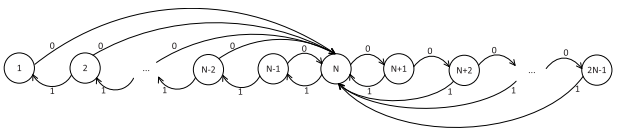

FIGURE 8. FSM of the binary pearson constraint.

State $N-1$ and $N+1$ are inferior to state 1 in terms of the maximum possible code rate [21], [22], [24], however, it can be verified that only words 01 and 101 are not synchronizing words in M_{N-1}, and the sync probability of M_{N-1} (and M_{N+1}) is 62.5%. Therefore, it is again observed that a tradeoff exists between the maximum possible code rate and the sync probability.

C. DC-FREE CONSTRAINTS

We consider DC-free constraints with N different RDS values, as depicted in the FSM shown in Fig. 9. According to [21], [22], state $\lceil N/2 \rceil$ should be selected as the specified state since its minimal set has the highest possible code rate. We first consider the case of the DC-free constraint with $N = 5$. We have that $M_3 = \{10, 01, 1100, 0011, 110100, 001011\}$ with $l_{max} = 6$. Unfortunately it can be verified that M_3 does not contain a synchronizing word, thus M_3 has a sync probability of 0%.

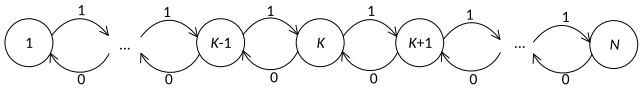

FIGURE 9. FSM of a DC-free constraint with N RDS values.

Now we consider the general case where $N > 5$. The detailed construction algorithm of minimal sets can be found in [22]. To illustrate, in Table 3 we list a minimal set when $N = 7$; this is Table 8 in [22]. Note that since state $\lceil N/2 \rceil$ has sequences 11 and 00 associated with both its outgoing and incoming edges, it does not satisfy Criterion 2 and thus may not be preferred in terms of sync probability. In fact, we can prove that with $N \geq 5$, the minimal set $M_{\lceil N/2 \rceil}$ has sync probability 0%. The proof is as follows.

TABLE 3. Words in the minimal set of a DC-free code with $N = 7$, $l_{max} = 10$, $\tilde{\mathfrak{C}}_M = 97.65\%$ (from [22]).

Word length	Word	Word length	Word
2	10	10	1110101000
2	01	10	0001010111
4	1100	10	1101101000
4	0011	10	0010010111
6	111000	10	1110011000
6	000111	10	0001100111
6	110100	10	1110100100
6	001011	10	0001011011
8	11101000	10	1101010100
8	00010111	10	0010101011
8	11011000	10	1101011000
8	00100111	10	0010100111
8	11100100	10	1101100100
8	00011011	10	0010011011
8	11010100	10	1110010100
8	00101011	10	0001101011

It is clear that words 01 and 10 are not synchronizing words since they violate Condition 1. Furthermore, we observe that

any of the remaining words $\mathcal{W}$ in $M_{\lceil N/2 \rceil}$ ends with 00 or 11. Therefore, $\mathcal{W}$ is a synchronizing word iff $\mathcal{W}\backslash 00$ is a valid word when 00 is the prefix of $\mathcal{W}$, and similarly iff $\mathcal{W}\backslash 11$ is a valid word when 11 is the prefix of $\mathcal{W}$. However, $\mathcal{W}\backslash 00$ cannot be a valid word since a valid word in $M_{\lceil N/2 \rceil}$ must have an equal number of zeros and ones, while $\mathcal{W}\backslash 00$ has two more ones than zeros. A similar argument can be made for $\mathcal{W}\backslash 11$. Hence none of the words in $M_{\lceil N/2 \rceil}$ is a synchronizing word and therefore the sync probability is 0%.

However, we note that state 1 does not violate Criterion 2, and that the minimal set associated with state 1 has a nonzero sync probability. For $N = 5$ and $l_{max} = 6$, $M_1 = \{10, 1100, 110100, 111000\}$, where it can be verified that 110100 and 111000 are synchronizing words, resulting in a sync probability of 7.4% which is equal to what is possible with M_5 and is higher than that can be achieved in other states.

We also note that with additional prior knowledge at the decoder, performance can be improved in terms of sync probability. For example, assume the decoder exploits its knowledge that all DC-free codewords are of even length. With $N = 5$, the sync probability of M_1 is improved since one more word, 1100, becomes a synchronizing word. In Section V we present more results assuming that the decoder has additional knowledge related to the constraint.

In this section we investigated the sync properties of a variety of constraints, and demonstrated that states that satisfy Criteria 1 and 2 can result in minimal sets with high sync probabilities. In the next section, we show that variable-length CS codes constructed via partial extensions of minimal sets can achieve both high efficiency and high sync probability.

IV. PARTIAL EXTENSIONS

Using words in the minimal set as the set of codewords may not result in capacity-approaching codes since higher efficiency is often achieved with larger codebooks. Therefore, we perform partial extensions over the minimal sets to generate larger codebooks. However, different from [20]–[26] where partial extensions are exhaustively performed and the one that has the highest efficiency is selected, in this section we introduce an algorithm that efficiently guides the partial extension process such that the resulting codebook has a high sync probability. Note that performing partial extensions without care can reduce the sync probability, as we show in the following example.

Example 5: ($d = 1, k = 3$ RLL constraint) As discussed in the previous section, we select state 1 *in Fig. 5 as the specified state that results in a minimal set of sync probability* 100%, *i.e.,* $M_1 = \{01, 001, 0001\}$. *If we extend word* 0001, *we have* $M_{1,p} = \{01, 001, 000101, 0001001, 00010001\}$ *where* 01, 001 *and* 00010001 *are not synchronizing words, resulting in a sync probability of only* 17%. *In contrast, extension of the words 01 and 001 results in sync probabilities of* 78% *and* 43% *respectively.*

This example motivates us to design a guided partial extension algorithm that aims to simultaneously achieve both high code efficiency and high sync probability.

A. EXTENDING SYNCHRONIZING VERSUS NONSYNCHRONIZING WORDS

We first consider whether to extend synchronizing words or nonsynchronizing words when our aim is to keep the sync probability high. We consider the following proposition and observation.

Proposition 1: Extending a synchronizing word lowers sync probability.

Proof: Consider a synchronizing word $\mathcal{W}$ in M. If we extend $\mathcal{W}$, the $|M|$ resulting words from extension of $\mathcal{W}$ cannot all be synchronizing words since one of the resulting words is $\mathcal{W}' = \mathcal{W} + \mathcal{W}$. $\mathcal{W}$ becomes a suffix after this extension, and $\mathcal{W}'\backslash\mathcal{W} = \mathcal{W}$ can no longer be a valid word. Therefore, $\mathcal{W}'$ is not a synchronizing word, which lowers the sum of maxentropic probabilities of all synchronizing words. □

Observation 3: Extending a nonsynchronizing word may increase the sync probability.

It is often the case that an extended word $\mathcal{W}'$ constructed through an extension of a nonsynchronizing word $\mathcal{W}$ is a synchronizing word, since $\mathcal{W}'$ consists of $\mathcal{W}$ concatenated with a valid word. $\mathcal{W}$ becomes the suffix of the extended word $\mathcal{W}'' = \mathcal{W} + \mathcal{W}$, hence any $\mathcal{W}' \neq \mathcal{W}''$ is likely to satisfy Condition 2.

For example, consider the ICI constraint in Fig. 7. $M_3 = \{0, 1, 2, 31, 300, 301, 302, 331, 332, 3300, 3301, 3302 \ldots\}$ where word 0 is not a synchronizing word. If we extend word 0, the resulting set is $\{\underbrace{00, 01, 02, 031, 0300, 0301,}_{\textit{partial extension}}$ $\underbrace{032, 0331, 0332, 03300, 03301, 03302,}_{\textit{partial extension}}$ 1, 2, 31, 300, 301, 302, 331, 332, 3300, 3301, 3302, $\ldots\}$ where only 00 is not a synchronizing word. Note that the extended words {01, 02, 031, 0300, 0301, 032, 0331, 0332, 03300, 03301, 03302} are synchronizing words, thus the sync probability has increased.

Based on Proposition 1 and Observation 3, we propose extending nonsynchronizing words whenever possible. Only when the minimal set does not contain a nonsynchronizing word do we recommend extending synchronizing words. Selection of synchronizing words that will be extended is discussed below.

B. THE GUIDED PARTIAL EXTENSION ALGORITHM

We start from the minimal set M, i.e., $M_p = M$. In each partial extension we first obtain the set of suffixes S where $\forall \mathcal{S} \in S$ is a suffix of a synchronizing word $\mathcal{W}$ in M_p, i.e., $\mathcal{W} = \zeta + \mathcal{S}$ where ζ denotes any suffix that exists in M_p. We search for a nonsynchronizing word $\mathcal{N}$ such that $\mathcal{N} \notin \mathcal{S}$ is the target word that we would like to extend. The reason for this is as follows. Suppose a synchronizing word $\mathcal{W}$ can be represented as $\mathcal{W} = \zeta + \mathcal{N}', \mathcal{N}' \in M_p \wedge S$. In this case, extending $\mathcal{N}'$ would make $\mathcal{W}$ nonsynchronizing since $\mathcal{N}'$ is not a valid word any more, which would reduce the sync probability. Therefore, we extend a nonsynchronizing word $\mathcal{N} \notin \mathcal{S}$. At the same time, $\mathcal{N}$ should not have a synchronizing word as its suffix,

i.e., $\mathcal{N} = \Lambda + \mathcal{W}$ (where Λ denotes any sequence) should not be extended since extension of $\mathcal{N}$ will result in $\mathcal{W}$ violating Condition 1 and becoming a nonsynchronizing word. If we cannot find such a word $\mathcal{N}$, then a partial extension will result in a synchronizing word becoming nonsynchronizing. In this case we choose to perform extension for each of the nonsynchronizing words and choose the one that results in the highest synchronization probability.

Data: words in M_p, the recursion depth J
Result: the updated M_p
Initialization: S, the set of nonsynchronizing words P, the set of synchronizing words Q, the set of words Γ that will be extended in the current recursion, $\Gamma = \Phi$
Sort(P) // *sort by length from shortest to longest*;
Sort(Q) // *sort by length from shortest to longest*;
if $P \neq \varnothing$ **then**
  foreach *word* $\mathcal{N} \in \mathcal{P}$ **do**
    if $\mathcal{N} \in \mathcal{S}$ *or* $\mathcal{N} = \mathcal{N}'' + \mathcal{W}$ $\exists \mathcal{W} \in Q$ **then**
      continue;
    else
      if Γ.*empty*() *or* $\mathcal{N}$.*size*() == Γ.*back*().*size*() **then**
        Γ.*push_back*($\mathcal{N}$);
      else
        break;
      end
    end
  end
  if Γ.*empty*() **then**
    construct Γ with all words in P;
  end
else
  construct Γ with the shortest words in Q;
end
/* *recursion with depth* $J + 1$ */;
foreach *word* $\gamma \in \Gamma$ **do**
  extend γ in M_p, obtain $M_{p,\gamma}$ and the corresponding synchronization probability; **call** *Algorithm 1* ($M_{p,\gamma}$, $J + 1$); undo the extension of γ in $M_{p,\gamma}$, backtrack to M_p;
end
Output: $M_{p,\gamma}$ that has the highest synchronization probability $\forall \gamma \in \Gamma$;

Algorithm 1 The Guided Partial Extension Algorithm

If there are no nonsynchronizing words in M_p, we must extend a synchronizing word. We note that extending a longer synchronizing word is more likely to reduce the sync probability than extending a shorter synchronizing word, since a longer synchronizing word $\mathcal{W}$ may correspond to a greater number of valid words $\mathcal{W}''$ where $\mathcal{W} = \Lambda + \mathcal{W}''$, and extension of $\mathcal{W}$ results in $\mathcal{W}''$ violating Condition 1. Returning to Example 5, it is straightforward to confirm that extension of word 0001 is not preferred since it results in words 01 and 001 violating Condition 1. On the other hand, extension of word 001 only excludes word 01 from being a synchronizing word, and extension of the word 01 does not result in any word violating Condition 1. Therefore, we choose to extend the shortest synchronizing word, and we propose Proposition 2 based on the following lemma.

Lemma 1: Under the condition that M_p does not contain nonsynchronizing words, the shortest synchronizing word cannot be represented as a suffix plus a valid word.

Proof: The proof is straightforward and is omitted. □

Proposition 2: Under the condition that M_p does not contain nonsynchronizing words, extending the shortest synchronizing word $\mathcal{W}$ reduces the sync probability by $\lambda_{max}^{-2l_{\mathcal{W}}}$ where $l_{\mathcal{W}}$ is the length of word $\mathcal{W}$.

Proof: Since $\mathcal{W}$ is a synchronizing word, it satisfies Condition 1. Therefore words resulting from extension of $\mathcal{W}$ also satisfy Condition 1. From Lemma 1 we know that these words also satisfy Condition 2 except for the extended word $\mathcal{W}'' = \mathcal{W} + \mathcal{W}$. It can be easily checked that all words in $M_p \backslash \mathcal{W}$ remain synchronizing words, because they satisfy both Conditions 1 and 2. The reduction of sync probability as the result of extending $\mathcal{W}$ is therefore the maxentropic probability of word $\mathcal{W}''$, which is $\lambda_{max}^{-2l_{\mathcal{W}}}$. □

Based on the above discussion, the proposed guided partial extension algorithm is shown in Algorithm 1. This algorithm is initialized with $M_p = M$ and $J = 0$ where J is the recursion depth, and is recursively called a number of times until J exceeds the pre-established limits. With each recursion depth the algorithm outputs a codebook with the highest synchronization probability at the current depth.

V. NUMERICAL RESULTS

In this section, we present results regarding the efficiency and sync probability of codes constructed based on the procedures outlined above. We evaluate, under the case of a binary symmetric channel (BSC), the sync properties in terms of the average number of bits and average number of codewords that the decoder requires to regain synchronization once synchronization is lost. We consider the BSC because a BSC is a general channel model that, with appropriate extension, can represent a wide range of scenarios where coded bits are corrupted in digital transmissions due to various factors such as additive noise, fading, interference, etc. The decoding algorithm that we consider is the conventional bit-by-bit decoding algorithm described in the Appendix of [26], and which is reproduced here as Algorithm 2.

A. UPPER BOUNDS OF THE AVERAGE NUMBER OF CODEWORDS AND BITS BEFORE RESYNCHRONIZATION

We first derive an upper bound on the number of codewords and the number of coded bits that are required for the decoder to regain synchronization once synchronization is lost. We denote the upper bound on the number of codewords N_c and the number of coded bits N_b as $\widetilde{N}_c$ and $\widetilde{N}_b$, respectively.

To evaluate $\widetilde{N}_c$, under the condition that there are no errors in the received symbols during synchronization, we consider the case when synchronization occurs only as a result of the

Data: the received sequence $\hat{v}$
Result: the decoded output, an estimation of the source sequence
Initialization: the codebook, l_{max}, $cur\_pos\_head \leftarrow 1$, $cur\_pos \leftarrow 1$
while $cur\_pos\_head \leq |\hat{v}|$ **do**
  if $cur\_pos - cur\_pos\_head + 1 \leq l_{max}$ **then**
    if $\hat{v}_{cur\_pos\_head}^{cur\_pos}$ *is a valid codeword* **then**
      decode $\hat{v}_{cur\_pos\_head}^{cur\_pos}$ into the corresponding source word; $cur\_pos \leftarrow cur\_pos + 1$;
      $cur\_pos\_head \leftarrow cur\_pos$;
    else
      $cur\_pos \leftarrow cur\_pos + 1$;
    end
  else
    /* *no match is found in the codebook* */;
    $cur\_pos \leftarrow cur\_pos\_head + 1$;
    while $cur\_pos \leq |\hat{v}|$ **do**
      foreach *tmp_start_pos in* $[cur\_pos\_head, cur\_pos]$ **do**
        if $\hat{v}_{tmp\_start\_pos}^{cur\_pos}$ *is a valid codeword* **then**
          decode $\hat{v}_{tmp\_start\_pos}^{cur\_pos}$;
          $cur\_pos \leftarrow cur\_pos + 1$;
          $cur\_pos\_head \leftarrow cur\_pos$; goto line 5;
        else
          $tmp\_start\_pos \leftarrow tmp\_start\_pos + 1$;
        end
      end
      $cur\_pos \leftarrow cur\_pos + 1$;
    end
  end
end

Algorithm 2 Conventional Bit-by-Bit Variable-Length CS Decoding When Errors Occur During Transmission (From [26])

occurrence of a sync word. We note that, after loss of synchronization, if the next received codeword is a synchronizing codeword (that occurs with probability P) then the receiver will regain synchronization. However, if the next codeword is not a synchronizing codeword (with probability $1-P$) but the subsequent word is a sync word, then synchronization will occur after two words with probability $(1-P)P$. Continuing, we have that

$$\widetilde{N}_c = \lim_{i\to\infty}\{P + 2(1-P)P + \ldots + i(1-P)^{i-1}P\} = \frac{1}{P}. \quad (7)$$

Now consider the case when errors on the binary symmetric channel occur with probability p_c. The probability that a codeword is correctly received is, on average, $(1-p_c)^{\bar{o}}$ where $\bar{o}$ denotes the average length of codewords, i.e., $\bar{o} = \sum_{s_i} 2^{-s_i} o_i$. Therefore, we have

$$\widetilde{N}_c = \frac{1}{P \times (1-p_c)^{\bar{o}}}, \quad (8)$$

In a similar fashion, $\widetilde{N}_b$ is derived as

$$\widetilde{N}_b = \frac{1}{P \times (1-p_c)^{\bar{o}}} \times \bar{o} + \bar{o} - 1 \quad (9)$$

where $\bar{o} - 1$ is, on average, the maximum number of bits of the currently received codeword that has caused mis-synchronization.

As noted above, in the derivation of (7) – (9), we assume that resynchronization occurs only on synchronizing codewords. However, it can be observed in Algorithm 2 that it is possible for resynchronization to also occur on nonsynchronizing codewords. Therefore, $\widetilde{N}_c$ and $\widetilde{N}_b$ are indeed upper bounds on the average number of codewords and the average number of bits that the receiver requires to regain synchronization, since the actual number of codewords the receiver requires to resynchronize can be smaller. As we will show in the simulation results, good synchronization properties can be observed even with a small value of P and correspondingly large values of $\widetilde{N}_c$ and $\widetilde{N}_b$.

B. SIMULATION RESULTS

We now consider simulation results for synchronization with several classes of constrained sequence codes.

1) RLL CONSTRAINTS

Consider, for example, the $(d = 1, k = 3)$ RLL constraint. In accordance with the discussion in Section III, we select state 1 as the specified state, hence $M_1 = \{01, 001, 0001\}$. We perform the guided partial extension algorithm over M_1 with J recursions according to Algorithm 1, $J = 0 \rightarrow 9$. The code efficiency and sync probability of the resulting codes are shown in Fig. 10, where $J = 0$ on the horizontal axis represents the code constructed with codewords from the minimal set. From this figure we can see that after several iterations of extension, we can construct codes with code efficiency near 99% and sync probability near 100%. The decrease at $J = 1$ is due to the fact that $P = \varnothing$ and we therefore have to extend a synchronizing word in the first extension, resulting in $M_{1,p} = \{001, 0001, 0101, 01001, 010001\}$ where 0101 is not a synchronizing word since it does not satisfy Condition 2.

For comparison purposes, consider the codebook that corresponds to $J = 4$ shown in Table 4. This code achieves $\eta = 98.90\%$ and $P = 96.88\%$. Note that this codebook has the same number of codewords and an efficiency very close to the code given in Table 2, however, the sync probability of the code in Table 2 is only 21.88%, which is much lower than the code in Table 4. This demonstrates that the guided partial extension algorithm can effectively generate codebooks with high sync probabilities.

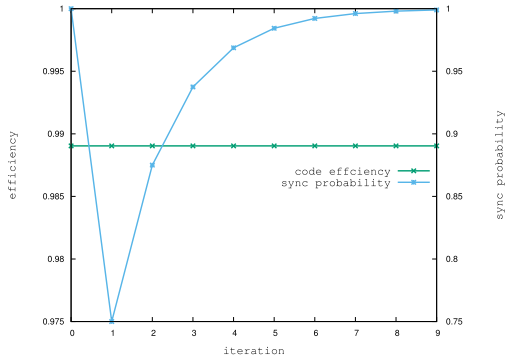

FIGURE 10. Code efficiency and sync probability of ($d = 1, k = 3$) RLL CS codes.

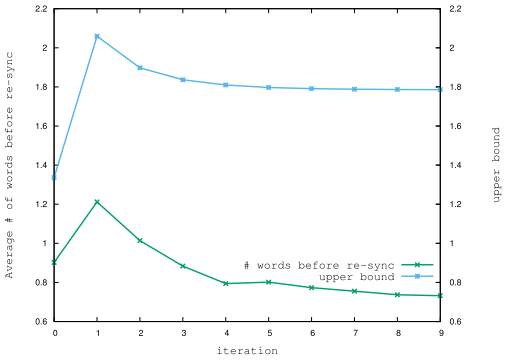

FIGURE 11. Average number of words required to regain synchronization for the constructed ($d = 1, k = 3$) RLL CS codes.

TABLE 4. Codebook of a ($d = 1, k = 3$) RLL code with efficiency of 98.90% and sync probability of 96.88%.

Source word	Codeword	Source word	Codeword
01	0001	00000	0101010001
11	001	00001	0101010101
001	010001	10001	010101001
101	01001	100000	010101010001
0001	01010001	100001	01010101001
1001	0101001		

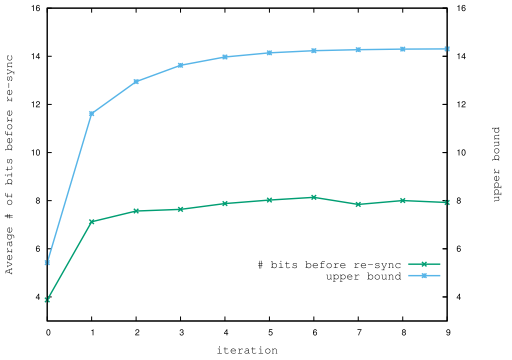

FIGURE 12. Average number of bits required to regain synchronization for the constructed ($d = 1, k = 3$) RLL CS codes.

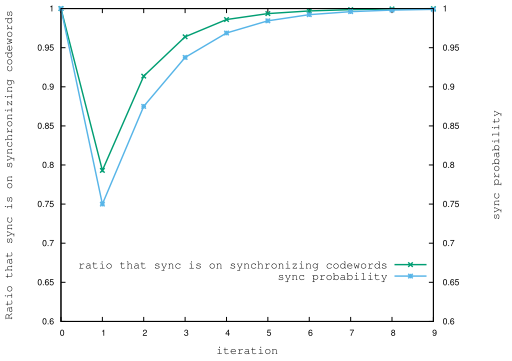

FIGURE 13. The ratio that synchronization is achieved on synchronizing codewords for the constructed ($d = 1, k = 3$) RLL CS codebooks.

We now consider the sync properties of the ($d = 1, k = 3$) RLL codes we constructed. The coded sequence is transmitted over a BSC with crossover probability 0.1. A source sequence of 50000 bits is randomly generated and encoded into a constrained sequence with the number of codewords ranging from ∼9000 to ∼18000 with $J = 0 \rightarrow 9$, according to the codebook. Once synchronization is lost due to errors that occur during transmission through simulation of Algorithm 2, we obtain the number of bits and the number of codewords before the receiver regains synchronization. We consider all the occurrences that synchronization is lost, we report the average number of bits and average number of codewords that the receiver receives before it re-synchronizes; the results are shown in Figs. 11 and 12. It can be seen that the receiver generally requires less than one codeword to regain synchronization, demonstrating that these codes have good synchronization properties in the sense that once synchronization is lost, they recover synchronization quickly. N_c first increases from $J = 0 \rightarrow 1$ and then decreases from $J = 1 \rightarrow 9$, which is consistent with the sync probability shown in Fig. 10. Fig. 12 shows that N_b is around 8 for $J = 1 \rightarrow 9$. This is because codebooks with larger J have longer codewords, therefore N_b does not reduce as dramatically as N_c. Note that with smaller crossover probabilities the receiver would need fewer codewords and coded bits to recover synchronization once synchronization is lost, and vice versa.

In Fig. 13 we demonstrate the ratio of the number of events when synchronization is achieved on synchronizing codewords to the total number of synchronization events, for $J = 0 \rightarrow 9$. As demonstrated in Fig. 13, these ratios are less than 100% indicating that some synchronization events occur on nonsynchronizing codewords. This demonstrates that Algorithm 2 permits synchronization to occur on nonsynchronizing codewords, as we mentioned above. This explains why in Figs. 11 and 12 the actual average number of codewords and average number of bits the receiver requires for resynchronization are lower than $\widetilde{N}_c$ and $\widetilde{N}_b$, since $\widetilde{N}_c$ and $\widetilde{N}_b$ assume that synchronization only occurs on synchronizing codewords.

Finally, as is clear from these figures, for the ($d = 1$, $k = 3$) RLL constraint there is no significant advantage

to using a codebook other than the minimal set because it satisfies Criteria 1 and 2, and hence has excellent sync properties, while also having high efficiency. In contrast, in the next subsection we examine situations in which $J = 0$ is not the best choice when we compare with other codebooks constructed using our guided partial extension algorithm.

2) FLASH MEMORIES

We consider constraints that mitigate ICI in flash memories, including the single-level cell (SLC) and multi-level cell (MLC) flash memories. For the SLC flash memories, the constraint was shown previously in Fig. 6; discussion in Section III-A reveals that it is sufficient to use M_1 as the minimal set. Therefore, we construct our codebooks based on state 1. The code efficiency and sync probability are shown in Fig. 14 where it can be seen that, similar to the situation with the $(d = 1, k = 3)$ RLL constraint, the sync probability decreases as $J = 0 \rightarrow 1$ since the synchronizing word 1 is extended, resulting in the nonsynchronizing word 11. For $J = 1 \rightarrow 9$, the sync probability increases up to 99.6%. The sync performance is shown in Figs. 15 and 16, where it is evident that on average less than one codeword and less than 8 bits are required for the receiver to regain synchronization.

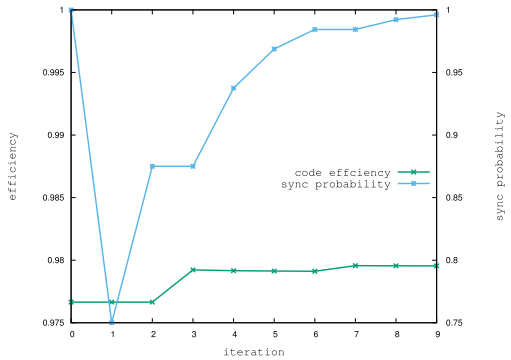

FIGURE 14. Code efficiency and sync probability of the codes for SLC flash memory.

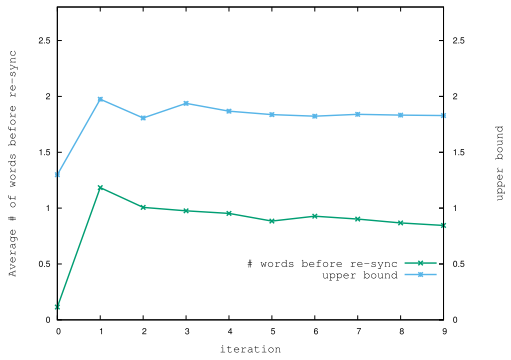

FIGURE 15. Average number of words required to regain synchronization for the codes for SLC flash memory.

Now we show that in situations where the minimal set has low sync probability, such as when the code efficiency is of high priority and the specified state is selected according to the criteria in [21], [22], the guided partial extension algorithm will likely help improve the sync probability. For example, we consider MLC flash memories and the FSM of the constraint that forbids pattern 303 as shown in Fig. 7. The capacity of this constraint is 1.978 bit/symbol. According to our criteria, state 1 has the best sync probability, but a lower code rate than states selected according to the criteria described in [21]–[25]. We instead consider selecting state 3 as the specified state which has the best maximum possible code rate, but worse sync probability compared to state 1. Note that state 3 does not satisfy Criterion 1, because word 0 is not a synchronizing word in M_3. If we directly perform NGH coding over M_3, the resulting codebook, shown in Table 5, achieves 99.6% of the capacity and has a sync probability of $P = 75\%$ since codeword 0, which occurs with 25% probability, is not a synchronizing codeword. We now show that the sync probability increases with the proposed guided partial extension algorithm.

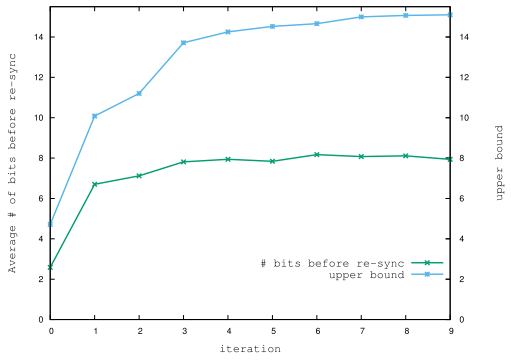

FIGURE 16. Average number of bits required to regain synchronization for the codes for SLC flash memories.

TABLE 5. A constrained sequence codebook for ICI mitigation of MLC flash memory that achieves 99.6% of capacity.

Source words	Codewords	Source words	Codewords
00	2	01	1
10	0	1100	32
1101	31	111000	302
111001	301	111010	300
111011	332	111100	331
1111010	3302	1111011	3301
1111100	3300	1111101	3332
11111100	3331	111111010	33302
111111011	33301	111111100	33300
111111101	33332	111111110	33331
11111111100	333302	11111111101	333301
111111111100	333300	111111111101	333332
111111111110	333331	1111111111110	3333302
11111111111110	3333301	11111111111111	3333300

Fig. 17 shows the code efficiency and sync probability of codebooks we have constructed for $J = 0 \rightarrow 9$. It can be seen that along with small increases in efficiency, the sync probability increases from 75% to 99.99%, which demonstrates the effectiveness of the proposed algorithm. Fig. 18 and 19 show the average number of codewords and

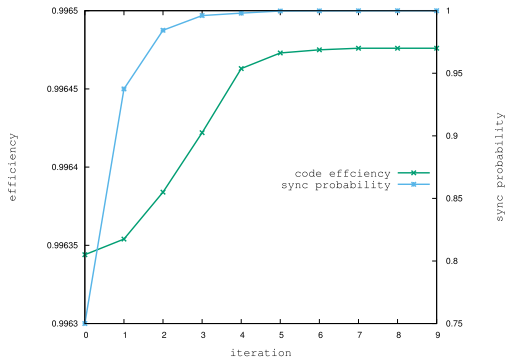

FIGURE 17. Code efficiency and sync probability of the constructed codes for MLC flash memory.

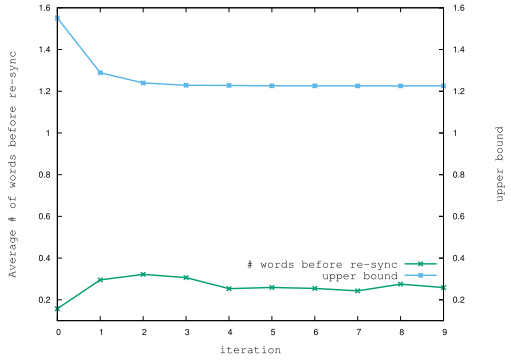

FIGURE 18. Average number of words required to regain synchronization for the constructed codes for MLC flash memory.

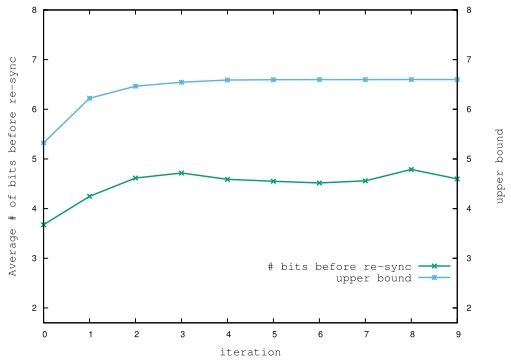

FIGURE 19. Average number of bits required to regain synchronization for the constructed codes for MLC flash memory.

the average number of bits that the decoder requires to receive to regain synchronization. It can be seen that on average, less than 0.4 codewords and less than 5 bits are needed to regain synchronization, which illustrates that the constructed codebooks have good synchronization properties. N_c is smaller than 1 because even when errors exist in the received bit sequence, Algorithm 2 usually correctly identifies the end of the current codeword.

3) THE PEARSON CONSTRAINT

With the Pearson constraint, we present the results with codes that are constructed using state N in Fig. 8 as the specified state. Fig. 20 shows the code efficiency and sync probability of the constructed codebooks with $J = 0 \rightarrow 9$. It can be seen that the sync probability increases from 50% to 61.5%. Figs. 21 and 22 show the average number of codewords and the average number of bits that the decoder needs to receive to regain synchronization. It can be seen that even though the sync probabilities are relatively low, on average approximately one codeword and fewer than 7 bits are needed to regain synchronization.

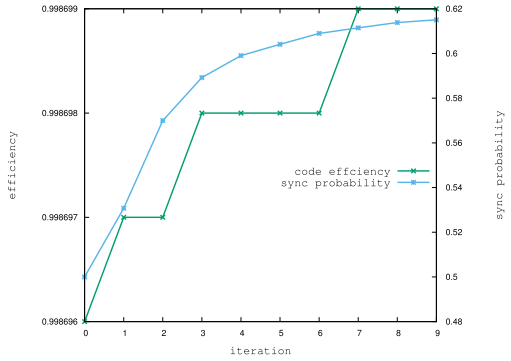

FIGURE 20. Code efficiency and sync probability of the pearson codes.

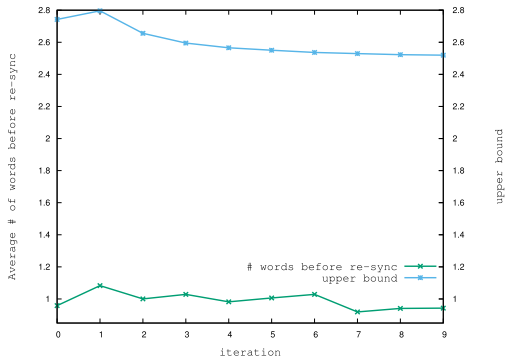

FIGURE 21. Average number of words required to regain synchronization for the constructed pearson codes.

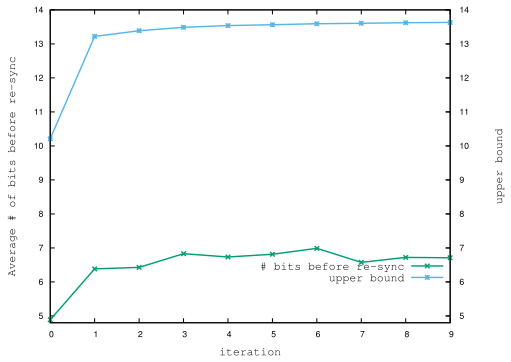

FIGURE 22. Average number of bits required to regain synchronization for the constructed pearson codes.

4) THE DC-FREE CONSTRAINT

We present the results of the DC-free constraint with $N = 5$, which has been adopted in visible light

communication systems for flicker reduction and dimming control [3], [46], [47], and consider the tradeoff between code efficiency and sync probability. According to the discussion in Section III-C, state 1 is better than state 2, which is better than state 3 in terms of sync probability. However, according to the study in [21]–[25], the opposite is true when attempting to maximize the code rate.

In Fig. 23 we present the result of code efficiency and sync probability for codes constructed with states 1, 2 and 3 as the specified state, with $J = 0 \rightarrow 9$. It can be seen that the abovementioned conclusion is verified, and the expected tradeoff between code efficiency and sync probability that arises from different states as the specified state is clearly seen. Figs. 24 and 25 show the average number of codewords and the average number of bits before the decoder regains synchronization. It can be seen that on average, between 2 to 7 codewords and between 10 to 35 bits are needed to regain synchronization. It is also demonstrated in Figs. 23–25 that state 1 is the best in terms of code efficiency while state 3 is the worst, but the opposite is observed in terms of synchronization properties. Therefore we once again observe the tradeoff between code efficiency and synchronization property. Note that $\widetilde{N}_c$ and $\widetilde{N}_b$ for states 2 and 3 are infinity since $P = 0$. However, because it is possible for the decoder to regain synchronization on nonsynchronizing codewords, these codes demonstrate relatively good synchronization properties even though $P = 0$.

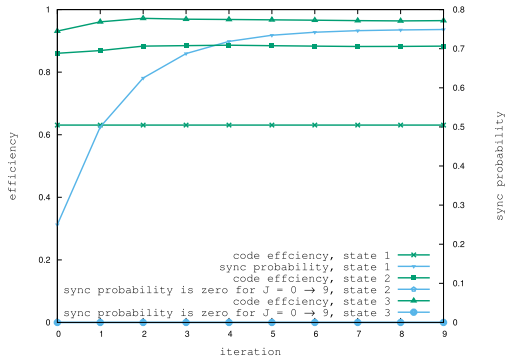

FIGURE 23. Code efficiency and sync probability of the constructed DC-free codes with $N = 5$.

To improve the synchronization properties, we use the prior knowledge that the codewords are of even length, and process two bits per decoding attempt. In this case, the sync probabilities are improved as shown in Fig. 26, and the average number of codewords and binary coded bits that are needed to regain synchronization are reduced as shown in Figs. 27 and 28. It can be seen that on average, approximately one codeword and fewer than 7 bits are needed to regain synchronization.

Note that in this case, the results do not indicate that the synchronization performance for state 1 > state 2 > state 3. The reason is illustrated in Fig. 29, and is explained as follows. Codeword 2 in the minimal set with state 1 as the specified state is not a synchronizing word since it does not

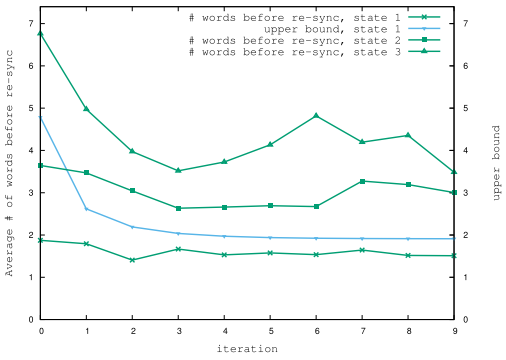

FIGURE 24. Average number of words required to regain synchronization for the constructed DC-free codes with $N = 5$. The upper bounds for states 2 and 3 are infinity since $P = 0$.

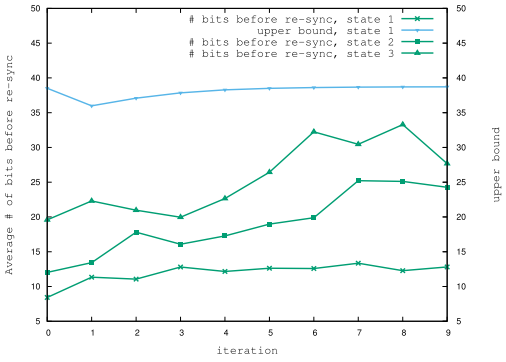

FIGURE 25. Average number of bits required to regain synchronization for the constructed DC-free codes with $N = 5$. The upper bounds for states 2 and 3 are infinity since $P = 0$.

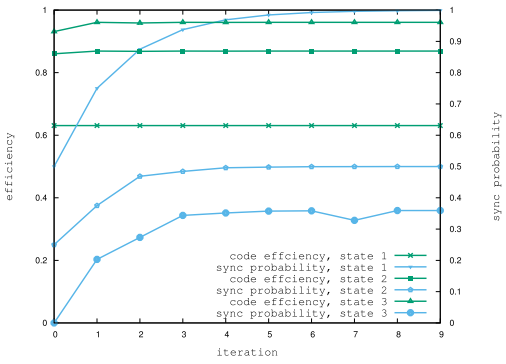

FIGURE 26. Code efficiency and sync probability of the constructed DC-free codes with $N = 5$, when the decoder has knowledge that codewords have even length.

satisfy Condition 1. However, the chances that reception of codeword 2 result in mis-synchronization is only when the quaternary bit before 2 (which can be 2 or 0) is incorrectly detected as 3 and the quaternary bit after 2 (which can be 2 or 3) is incorrectly detected as 0. The probability of this case is low, hence codeword 2 can be regarded as an "almost synchronizing codeword". It follows that the sync probability of the minimal set of state 1 can be regarded as "almost 100%". Similar reasoning holds for states 2 and 3, making their sync

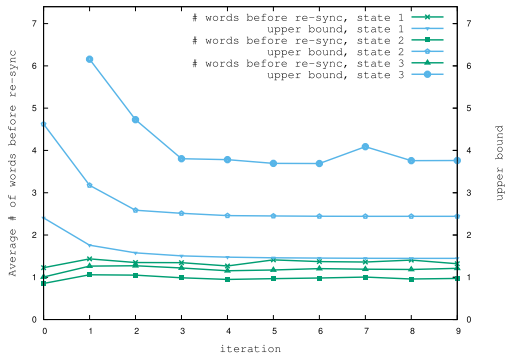

FIGURE 27. Average number of words required to regain synchronization for the constructed DC-free codes with $N = 5$, when the decoder has knowledge that codewords have even length.

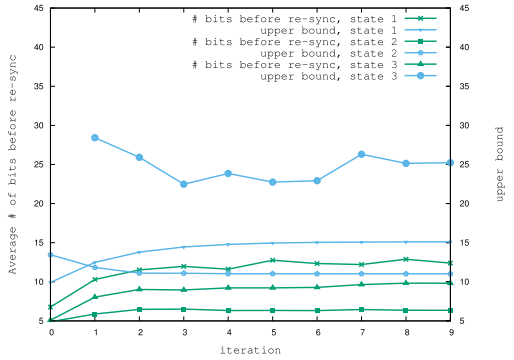

FIGURE 28. Average number of bits required to regain synchronization for the constructed DC-free codes with $N = 5$, when the decoder has knowledge that codewords have even length.

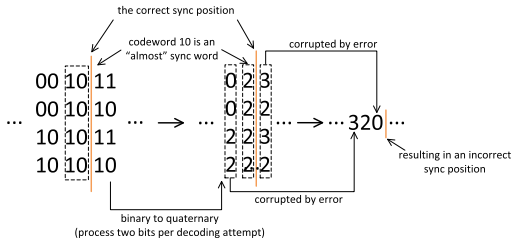

FIGURE 29. All cases that codeword 2 in the minimal set constructed based on state 1 results in mis-synchronization. Note that the probabilities of these cases are small, making codeword 2 an "almost synchronizing codeword".

probability "almost 100%" , and the number of codewords that are needed to regain synchronization is similar for all three states.

We also note that, with prior knowledge that all codewords have even length, in case that the receiver misses a single bit in the detection process, the above-mentioned decoding process with two bits per decoding attempt will never re-synchronize. Therefore, we propose starting the two-bit-grouping at both odd and even positions and performing decoding with both alternatives. In situations where the receiver misses a bit or mistakenly clocks in an extra bit, the decoding attempt that starts at odd positions will re-synchronize, otherwise the decoding attempt that starts at even positions will re-synchronize the received sequence.

VI. CONCLUSION

In this paper, we have discussed initially establishing and re-establishing codeword boundaries in the decoding of variable-length constrained sequence codes. We studied criteria for selection of the specified state in the minimal set that achieves a high synchronization probability, and showed that these criteria can lead to selection of different specified states compared to the criteria developed in [21], [22] that aimed to maximize code rate. We then proposed the guided partial extension algorithm to increase code efficiency while maintaining high sync probability. Finally, we presented simulation results that demonstrate the code efficiency and sync probability of the codes we constructed; these results included the average number of received codewords and the average number of coded bits that the decoder requires to regain synchronization should synchronization be lost. We demonstrated that it is possible to construct highly efficient variable-length CS codes that exhibit good synchronization properties such that very few codewords and very few bits are required to regain synchronization.

REFERENCES

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

[2] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, "Energy harvesting wireless communications: A review of recent advances," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 360–381, Mar. 2015.

[3] *IEEE Standard for Local and Metropolitan Area Networks—Part 15.7: Short-Range Wireless Optical Communication Using Visible Light*, Standard 802.15.7, 2011, pp. 248–271.

[4] K. A. Schouhamer Immink and K. Cai, "Design of capacity-approaching constrained codes for DNA-based storage systems," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 224–227, Feb. 2018.

[5] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[6] P. A. Franaszek, "Sequence-state encoding for digital transmission," *Bell Syst. Tech. J.*, vol. 47, pp. 143–157, Jan. 1968.

[7] A. X. Widmer and P. A. Franaszek, "A DC-balanced, partitioned-block, 8B/10B transmission code," *IBM J. Res. Develop.*, vol. 27, no. 5, pp. 440–451, Sep. 1983.

[8] K. A. S. Immink, J.-Y. Kim, S.-W. Suh, and S. Keun Ahn, "Efficient DC-free RLL codes for optical recording," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 326–331, Mar. 2003.

[9] C. Jamieson and I. Fair, "Construction of constrained codes for state-independent decoding," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 193–199, Feb. 2010.

[10] R. Motwani, "Hierarchical constrained coding for floating-gate to floating-gate coupling mitigation in flash memory," in *Proc. IEEE Global Telecommun. Conf. - GLOBECOM*, Dec. 2011, pp. 1–5.

[11] H. Zhou, A. Jiang, and J. Bruck, "Balanced modulation for nonvolatile memories," 2012, *arXiv:1209.0744*. [Online]. Available: http://arxiv.org/abs/1209.0744

[12] F. R. Kschischang and T. Lutz, "A constrained coding approach to error-free half-duplex relay networks," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6258–6260, Oct. 2013.

[13] K. A. S. Immink and J. H. Weber, "Minimum pearson distance detection for multilevel channels with gain and/or offset mismatch," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5966–5974, Oct. 2014.

[14] J. H. Weber, K. A. S. Immink, and S. R. Blackburn, "Pearson codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 131–135, Jan. 2016.

[15] K. Schouhamer Immink and V. Skachek, "Minimum pearson distance detection using mass-centered codewords in the presence of unknown varying offset," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2510–2517, Sep. 2016.
[16] C. Cao, D. Li, and I. Fair, "Deep learning-based decoding for constrained sequence codes," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–7.
[17] K. A. S. Immink and J. H. Weber, "Very efficient balanced codes," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 188–192, Feb. 2010.
[18] J. H. Weber, T. G. Swart, and K. A. Schouhamer Immink, "Simple systematic pearson coding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 385–389.
[19] T. G. Swart and J. H. Weber, "Binary variable-to-fixed length balancing scheme with simple encoding/decoding," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 1992–1995, Oct. 2018.
[20] A. Steadman and I. Fair, "Variable-length constrained sequence codes," *IEEE Commun. Lett.*, vol. 17, no. 1, pp. 139–142, Jan. 2013.
[21] C. Cao and I. Fair, "Construction of minimal sets for capacity-approaching variable-length constrained sequence codes," in *Proc. 50th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2016, pp. 255–259.
[22] C. Cao and I. Fair, "Minimal sets for capacity-approaching variable-length constrained sequence codes," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 890–902, Feb. 2019.
[23] C. Cao and I. Fair, "Mitigation of inter-cell interference in flash memory with capacity-approaching variable-length constrained sequence codes," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2366–2377, Sep. 2016.
[24] C. Cao and I. Fair, "Capacity-approaching variable-length pearson codes," *IEEE Commun. Lett.*, vol. 22, no. 7, pp. 1310–1313, Jul. 2018.
[25] C. Cao and I. Fair, "Construction of multi-state capacity-approaching variable-length constrained sequence codes with state-independent decoding," *IEEE Access*, vol. 7, pp. 54746–54759, 2019.
[26] C. Cao, D. Li, and I. Fair, "Deep learning-based decoding of constrained sequence codes," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2532–2543, Nov. 2019.
[27] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.
[28] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
[29] O. Dizdar and E. Arıkan, "A high-throughput energy-efficient implementation of successive cancellation decoder for polar codes using combinational logic," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 3, pp. 436–447, Mar. 2016.
[30] C. Cao, T. Koike-Akino, Y. Wang, and S. C. Draper, "Irregular polar coding for massive MIMO channels," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.
[31] T. Koike-Akino, C. Cao, Y. Wang, S. C. Draper, D. S. Millar, K. Kojima, K. Parsons, L. Galdino, D. J. Elson, D. Lavery, and P. Bayvel, "Irregular polar coding for complexity-constrained lightwave systems," *J. Lightw. Technol.*, vol. 36, no. 11, pp. 2248–2258, Jun. 1, 2018.
[32] L. M. Zhang and F. R. Kschischang, "Staircase codes with 6% to 33% overhead," *J. Lightw. Technol.*, vol. 32, no. 10, pp. 1999–2002, May 2014.
[33] M. Barakatain and F. R. Kschischang, "Low-complexity concatenated LDPC-staircase codes," *J. Lightw. Technol.*, vol. 36, no. 12, pp. 2443–2449, Jun. 15, 2018.
[34] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, Aug. 1998.
[35] T. Koike-Akino, C. Cao, and Y. Wang, "Turbo product codes with irregular polar coding for high-throughput parallel decoding in wireless OFDM transmission," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.
[36] E. Hemo and Y. Cassuto, "d-imbalance WOM codes for reduced inter-cell interference in multi-level NVMs," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2378–2390, Sep. 2016.
[37] V. Taranalli, H. Uchikawa, and P. H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 271–276.
[38] Y. M. Chee, C. Johan, H. M. Kiah, S. Ling, T. T. Nguyen, and V. K. Vu, "Efficient encoding/decoding of capacity-achieving constant-composition ICI-free codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 205–209.
[39] S. Buzaglo and P. H. Siegel, "Row-by-row coding schemes for inter-cell interference in flash memory," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4101–4113, Oct. 2017.
[40] K. A. S. Immink, "Some statistical properties of maxentropic runlength-limited sequences," *Philips J. Res.*, vol. 38, no. 3, pp. 138–149, 1983.
[41] G. Bocherer and R. Mathar, "Matching dyadic distributions to channels," in *Proc. Data Compress. Conf.*, Mar. 2011, pp. 23–32.
[42] G. Böcherer, "Capacity-achieving probabilistic shaping for noisy and noiseless channels," Ph.D. dissertation, Inst. Theor. Inf. Technol., RWTH Aachen Univ., Aachen, Germany, 2012. [Online]. Available: http://www.georg-boecherer.de/capacityAchievingShaping.pdf
[43] G. Böcherer. (Dec. 2010). *Geometric Huffman Coding*. [Online]. Available: http://www.georg-boecherer.de/ghc
[44] T. Ferguson and J. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 4, pp. 687–693, Jul. 1984.
[45] K. A. S. Immink, "Coding schemes for multi-level flash memories that are intrinsically resistant against unknown gain and/or offset using reference symbols," *Electron. Lett.*, vol. 50, no. 1, pp. 20–22, Jan. 2014.
[46] S. Rajagopal, R. Roberts, and S.-K. Lim, "IEEE 802.15.7 visible light communication: Modulation schemes and dimming support," *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 72–82, Mar. 2012.
[47] C. E. Mejia, C. N. Georghiades, M. M. Abdallah, and Y. H. Al-Badarneh, "Code design for flicker mitigation in visible light communications using finite state machines," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 2091–2100, May 2017.

CONGZHE CAO received the B.Eng. degree from the University of Science and Technology Beijing, in 2012, the M.Eng. degree in communications engineering from the Beijing Institute of Technology, Beijing, China, in 2014, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Alberta, Canada, in 2019. From 2016 to 2017, he was a Research Intern with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he was involved in designing advanced coding techniques for the next-generation wireless and fiber optic communication systems. He is currently a Senior Compiler Software Engineer with Huawei, Canada. His research interests include constrained sequence coding and error correction coding for digital communication and emerging data storage systems, information theory, machine learning, deep learning, compiler optimizations, and computer architecture.

IVAN FAIR (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from the University of Alberta, and the Ph.D. degree in electrical and computer engineering from the University of Victoria. He designed and implemented various aspects of communication systems with Bell Northern Research Ltd., and MPR TelTech Ltd., prior to joining the Technical University of Nova Scotia (since amalgamated with Dalhousie University) as a Faculty Member. He is currently a Professor with the University of Alberta, where he has served in several administrative roles, where he is also the Chair of the Department of Electrical and Computer Engineering.