Making WOM Codes Decodable Using Short Synchronous WOM Codes

Nicolas Bitouzé[†], Alexandre Graell i Amat[‡], and Eirik Rosnes[§]

[†]Department of Electronics, Telecom Bretagne, Brest, France
[‡]Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden
[§]Department of Informatics, University of Bergen, N-5020 Bergen, Norway
nicolas.bitouze@telecom-bretagne.eu, alexandre.graell@chalmers.se, eirik@ii.uib.no

Abstract—**While some write once memory (WOM) codes are inherently decodable, others require the added knowledge of the current generation in order to successfully decode the state of the memory. If there is no limit on the code length, n, a binary non-decodable t-write WOM code can be made decodable at an insignificant cost in terms of code rate by adding $t-1$ cells to store the current generation after replicating the code enough times for the $t-1$ cells to be of negligible weight. This justifies the research on non-decodable WOM codes. However, if n is bounded, the $t-1$ additional cells may introduce a significant loss in terms of code rate. In this paper, we propose a new method to make non-decodable WOM codes decodable at a lower price when n is bounded. The main idea is to add cells that do not only store the current generation, but also additional data, by using a *synchronous* $(t-1)$-write WOM code of length $t-1$ or slightly above which does not contain the all-zero codeword. A bound on the rate of a simple family of synchronous WOM codes with $n=t$ is given, as well as very short codes from this family. Better codes are then obtained by local manipulations of these codes. Finally, a construction of synchronous WOM codes with good properties is proposed to reach higher values of t.**

I. INTRODUCTION AND DEFINITIONS

A write once memory (WOM) [1] is a storage device consisting of memory cells that take on $q \geq 2$ possible states in $\{0, \ldots, q-1\}$, and such that the state of a given cell cannot be decreased. The main problem in the WOM model is to know how much information can be stored into n q-ary memory cells using t writes (also called *generations*), starting from the all-zero state. Formally, we are looking for t-write WOM codes, which are codes designed to store and update data in the WOMs using t writes. WOM codes are defined by their t encoding and decoding maps. The following definition is taken from [2]:

Definition 1: An $[n, t : M_1, \ldots, M_t]$ t-write q-ary WOM code C is a coding scheme for n q-ary WOM cells, which consists of t pairs of encoding and decoding maps $\mathcal{E}_i$ and $\mathcal{D}_i$ ($1 \leq i \leq t$) such that:

1) $\mathcal{E}_1 : \{1, \ldots, M_1\} \to \{0, \ldots, q-1\}^n$.
2) For $2 \leq i \leq t$:
   - $\mathcal{E}_i : \{1, \ldots, M_i\} \times Im(\mathcal{E}_{i-1}) \to \{0, \ldots, q-1\}^n$,
   - $\forall (m, \mathbf{c}) \in \{1, \ldots, M_i\} \times Im(\mathcal{E}_{i-1})$,
     $\forall j \in \{1, \ldots, n\}, (\mathcal{E}_i(m, \mathbf{c}))_j \geq (\mathbf{c})_j$.

3) For $1 \leq i \leq t$, $\mathcal{D}_i : \{0, \ldots, q-1\}^n \to \{1, \ldots, M_i\}$, and
   - $\forall m \in \{1, \ldots, M_1\}, \mathcal{D}_1(\mathcal{E}_1(m)) = m$,
   - for $2 \leq i \leq t$, $\forall (m, \mathbf{c}) \in \{1, \ldots, M_i\} \times Im(\mathcal{E}_{i-1})$,
     $\mathcal{D}_i(\mathcal{E}_i(m, \mathbf{c})) = m$.

For simplicity, in the remainder of the paper, we will refer to WOM codes simply as codes. The rate of the above code, referred to as the WOM-rate, is defined as follows [2]:

Definition 2: The rate of generation $i \in \{1, \ldots, t\}$ of an $[n, t : M_1, \ldots, M_t]$ q-ary code C is

$$\mathcal{R}_i(C) \triangleq \frac{\log_q M_i}{n} \tag{1}$$

and the WOM-rate of C is defined as

$$\mathcal{R}(C) \triangleq \sum_{i=1}^{t} \mathcal{R}_i(C) = \frac{\sum_{i=1}^{t} \log_q M_i}{n}. \tag{2}$$

Given t, q and sometimes n, one would like to maximize the WOM-rate. In this paper, we only consider binary codes, i.e., $q = 2$.

Depending on the structure of the code, the state of the cells may or may not suffice to determine the current generation (i.e., how many times the block has been written, and which map $\mathcal{D}_i$ should be used to decode). In other words, the code is not always decodable. We call *decodable* codes the codes such that for any state of the cells $\mathbf{c}$ and any i_1 and i_2 with $\mathbf{c} \in Im(\mathcal{E}_{i_1}) \cap Im(\mathcal{E}_{i_2})$, $\mathcal{D}_{i_1}(\mathbf{c}) = \mathcal{D}_{i_2}(\mathbf{c})$. A code is called *synchronous* [1] if a given state of the cells can only be reached at a given generation, i.e., the sets $Im(\mathcal{E}_i)$ are disjoint for $1 \leq i \leq t$. A simple way to guarantee that a code is synchronous is to force the Hamming weight w of the cells to be an injective function of the generation, i.e., for $\mathbf{c}_1 \in Im(\mathcal{E}_{i_1})$ and $\mathbf{c}_2 \in Im(\mathcal{E}_{i_2})$, $w(\mathbf{c}_1) = w(\mathbf{c}_2) \Rightarrow i_1 = i_2$. These codes are called *laminar* in [1]. By construction, synchronous codes are decodable. A construction of synchronous (and laminar) codes was given in [1] for $n = t$ being a power of two, and WOM-rate $\log_2(t)/2$. However, synchronous codes have not been extensively studied. Non-synchronous codes can still be directly decoded if, when the decoder cannot determine the current generation, the choice of $\mathcal{D}_i$ has no impact on the decoded symbol. Notice that synchronous codes are decodable, but the reverse does not always hold. For later use, if an $[n, t :$

$M_1, \ldots, M_t]$ code is synchronous (respectively decodable), we will use the superscript "sync", $[n, t : M_1, \ldots, M_t]^{\text{sync}}$ (respectively "dec", $[n, t : M_1, \ldots, M_t]^{\text{dec}}$). Also, the binary cells that can be written from 0 to 1 but not from 1 to 0 are called *wits* [1].

A non-decodable code C with parameters $[n, t : M_1, \ldots, M_t]$ can be turned into a decodable (and even synchronous) code by simply concatenating k instances of C with a block of $t - 1$ cells that stores the current generation (by being filled one by one at each write, starting at the second generation). The resulting code is a synchronous code with parameters $[kn + t - 1, t : M_1^k, \ldots, M_t^k]$. Note that if k goes to infinity, the WOM-rate of this code goes to the WOM-rate of the original code C, $\mathcal{R}(C)$. A common approach in the literature is to design codes that approach the boundaries of the capacity region (see, e.g., [2], [3]), and then make them decodable using this method. Therefore, most of the state-of-the-art high-rate codes are not *directly* decodable. However, if the target application specifies t and n, making a non-decodable code decodable using the above-mentioned method can significantly degrade its WOM-rate. For instance, consider $n = 6$ and $t = 4$, and assume that we do not know a decodable code of length 6. In this case, we could select a non-decodable 4-write code of length 3, and append 3 cells to store the current generation. The resulting WOM-rate is half the original one, as the additional cells only carry information about the current generation.

Notice that if the system must be able to know, when the state of the memory is the all-zero vector, whether this is because the block is empty or because it contains the all-zero codeword, then adding $t - 1$ cells is not enough to make the code decodable, but t cells are required in this case, instead. Here we consider the case where $t - 1$ additional cells are enough. The analysis for the other scenario is extremely similar.

In this paper, we propose a different approach to make a non-decodable t-write code C decodable. The key idea is to append $t - 1$ additional cells which store not only the current generation but also new data, by using a t-write synchronous code with length $t - 1$, and writing generations of C and of the synchronous code simultaneously. In the scenario where the system must know the difference between the all-zero codeword and an empty block of memory, a t-write synchronous code of length t which does not contain the all-zero codeword would be required. To unify the search for codes for both scenarios, we search for codes with $n = t$ which do not contain the all-zero codeword, and we turn them into a code suited for the scenario where the distinction between the all-zero codeword and an empty block is not required, by adding a generation that only contains the all-zero codeword. Then, synchronousness guarantees that by observing the $t - 1$ new cells, the decoder can always determine the current generation, and use this knowledge to decode the obtained code. We also consider using t-write synchronous codes with length slightly above $t - 1$ (the length should remain small, because we do not expect to find synchronous codes of

WOM-rate higher than non-decodable ones, thus the highest number of cells should be reserved to the non-decodable code). In particular, our focus is on binary laminar codes, but the proposed approach can be extended to non-laminar codes and non-binary codes.

II. LAMINAR WOM CODES WITH $n = t$

In this section, we focus on building laminar codes with $n = t$, that write exactly 1 wit at each generation. Also, in order to simplify the problem, we try to maximize the values of M_i generation by generation, rather than globally maximizing the WOM-rate. Consider a code C with $n = t$ that writes exactly 1 wit per generation, and a generation $i > 1$. Assuming that the previous generations are already fixed, the condition we have on M_i is that for every $\mathbf{x} \in Im(\mathcal{E}_{i-1})$, and for every $m \in \{1, \ldots, M_i\}$, there exists $\mathbf{y} \in Im(\mathcal{E}_i)$ such that $\mathbf{x} \leq \mathbf{y}$ and $\mathcal{D}_i(\mathbf{y}) = m$ (where $\mathbf{x} \leq \mathbf{y}$ if $x_k \leq y_k$ for all k, $1 \leq k \leq n$). Denote by E_i^n the set of binary vectors of length n and Hamming weight i. It follows that at each generation i, $Im(\mathcal{E}_i) \subseteq E_i^n$. We use this set inclusion to make our maximization at each generation completely independent from the other generations, at the cost of optimality.

Let us define the equivalence relation $\equiv_i^n$ on $Im(\mathcal{E}_i)$ by $\mathbf{y} \equiv_i^n \mathbf{y}'$ if and only if $\mathcal{D}_i(\mathbf{y}) = \mathcal{D}_i(\mathbf{y}')$. Let us refer to the equivalence classes of this relation as the *codeword classes* of C at generation i. Codeword classes are subsets $Y \subseteq E_i^n$ for which, if we do not take the previous generations into account, the following must hold:

$$\forall \mathbf{x} \in E_{i-1}^n, \exists \mathbf{y} \in Y : \mathbf{x} \leq \mathbf{y}. \tag{3}$$

We are also interested in the partitions of E_i^n as a set of valid codeword classes. If $\mathcal{Y}$ denotes such a partition, we want that

$$\forall Y \in \mathcal{Y}, \forall \mathbf{x} \in E_{i-1}^n, \exists \mathbf{y} \in Y : \mathbf{x} \leq \mathbf{y}. \tag{4}$$

Each valid partition $\mathcal{Y}$ corresponds to a valid decoding map (modulo reordering), and thus each cardinality $|\mathcal{Y}|$ to a valid M_i. We are therefore interested in finding the maximum cardinality, denoted by $A_i(n)$, of such a partition. We give an upper bound on $A_i(n)$:

Proposition 1: Let $B_i(n)$ be defined by

$$B_i(n) \triangleq \left\lfloor \frac{\binom{n}{i}}{\min_{Y \text{ s.t. (3) holds}} |Y|} \right\rfloor. \tag{5}$$

Then, the maximum cardinality $A_i(n)$ of a partition $\mathcal{Y}$ that satisfies (4) is upper bounded by $A_i(n) \leq B_i(n)$.

Proof: Let $\mathcal{Y}$ be any partition of E_i^n. Then,

$$|\mathcal{Y}| \cdot \left(\min_{Y \text{ s.t. (3) holds}} |Y| \right) \leq \sum_{Y \in \mathcal{Y}} |Y| = |E_i^n| = \binom{n}{i}. \tag{6}$$

This holds in particular when $\mathcal{Y}$ is of maximum cardinality. $\square$

This bound can be computed using a computer search for the smallest Y that satisfies (3). The search is relatively slow, but notice that by lower-bounding $|Y|$ by $\frac{|E_{i-1}^n|}{i}$ (each element

n \ i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**1**															
2	**2**	**1**														
3	**3**	**1**	**1**													
4	**4**	**3**	**1**	**1**												
5	**5**	**3**	**2**	**1**	**1**											
6	**6**	**5**	3	**2**	**1**	**1**										
7	**7**	**5**	**5**	2	2	**1**	**1**									
8	**8**	7	5	5	2	2	**1**	**1**								
9	**9**	7	6	5	3	2	2	**1**	**1**							
10	**10**	9	6	5	4	3	2	2	**1**	**1**						
11	**11**	9	7	6	5	4	3	2	2	**1**	**1**					
12	**12**	11	8	6	6	5	3	3	2	1	**1**	**1**				
13	**13**	11	10	7	6	5	4	3	3	2	2	**1**	**1**			
14	**14**	13	10	9	7	6	5	5	3	3	2	2	**1**	**1**		
15	**15**	13	13	9	9	6	5	5	4	3	3	2	2	**1**	**1**	
16	**16**	15	13	13	9	9	7	6	5	4	3	3	2	2	**1**	**1**

$\mathbf{y} \in E_i^n$ covers exactly i elements $\mathbf{x} \in E_{i-1}^n$), we obtain a closed-form bound:

$$A_i(n) \le B_i(n) \le \left\lfloor \frac{\binom{n}{i}}{\left\lceil \frac{|E_{i-1}^n|}{i} \right\rceil} \right\rfloor = \left\lfloor \frac{\binom{n}{i}}{\left\lceil \frac{\binom{n}{i-1}}{i} \right\rceil} \right\rfloor. \quad (7)$$

While the closed-form bound can be computed efficiently and is reached for some values of (n, i) (for instance, for $n \le 3$, or for $i \le 2$, or $i = n$), even for relatively low values of n and i, it can be strictly higher than $A_i(n)$. For instance, $A_3(4) = 1$, while the closed-form bound is 2. Indeed, $E_3^4 = \{1110, 1101, 1011, 0111\}$ and $E_2^4 = \{1100, 1010, 1001, 0110, 0101, 0011\}$, and while each element of E_3^4 covers 3 elements of E_2^4, it is not possible to pick two elements of E_3^4 that cover distinct elements of E_2^4. Therefore, the codeword classes in E_3^4 have cardinality at least 3, and not $\frac{|E_{i-1}^n|}{i} = 2$.

A. Comparison with a Computer Search for Small n

For very small values of n, the exact value of $A_i(n)$ can be computed by conducting a simple exhaustive search on the set of codeword classes. Values of $B_i(n)$ are also obtained with an exhaustive search, but on the minimum size of codeword classes, which is significantly faster. The results of the two searches are reported for $n \le 16$ in Table I. The values in bold font are $A_i(n)$, the others are $B_i(n)$. The few values of $A_i(n)$ that were computed exactly match $B_i(n)$, so it is unknown whether there are pairs (n, i) such that $A_i(n) < B_i(n)$. Note that these values are constructive. For instance, a $[4, 4 : 4, 3, 1, 1]^{\mathrm{sync}}$ and a $[5, 5 : 5, 3, 2, 1, 1]^{\mathrm{sync}}$ code can be obtained from the search.

III. LAMINAR WOM CODES WITH $n > t$

The constraints that we applied on the codes of Section II, especially the constraint that $n = t$, keep the code WOM-rates relatively low. Lifting the constraint on $n = t$ allows for higher WOM-rates, and laminar codes with n slightly larger than t can easily be derived from codes obtained as

in Section II by merging several generations together: taking, as the new set of codeword classes, the union of the sets of codeword classes of two or more consecutive generations. For instance, the $[4, 4 : 4, 3, 1, 1]^{\mathrm{sync}}$ code can be turned into a $[4, 3 : 4, 3, 2]^{\mathrm{sync}}$ code by merging its third and fourth generations together. Instead of having one codeword class at generation 3 ($\{1110, 1101, 1011, 0111\}$) and one at generation 4 ($\{1111\}$), now the third generation has two codeword classes: $\{1110, 1101, 1011, 0111\}$ and $\{1111\}$, and there is no fourth generation anymore. Likewise, a $[5, 3 : 5, 3, 4]^{\mathrm{sync}}$ code (of WOM-rate 1.181) can be derived from the $[5, 5 : 5, 3, 2, 1, 1]^{\mathrm{sync}}$ code by merging the last three generations together. However, consider the codeword classes of vectors of weight 4. These were constructed in order to cover every word of weight 3, while they now only have to cover every word of weight 2. The optimization also did not allow codeword classes of mixed weights. We can reorganize the set of vectors of weight 3 or more into a better balanced set of codeword classes. In (8), we give the codeword classes of the third generation of a $[5, 3 : 5, 3, 6]^{\mathrm{sync}}$ code (of WOM-rate 1.298) obtained by reorganizing the third generation of the $[5, 3 : 5, 3, 4]^{\mathrm{sync}}$ code:

$$\{01111, 11001, 10110\}, \{10111, 11100, 01011\},$$
$$\{11011, 01110, 10101\}, \{11101, 00111, 11010\},$$
$$\{11110, 10011, 01101\}, \{11111\}. \quad (8)$$

For comparison, the 4 codeword classes of the third generation of the $[5, 3 : 5, 3, 4]^{\mathrm{sync}}$ code are:

$$\{11100, 11010, 10101, 01011, 00111\} \text{ (weight 3 only)},$$
$$\{11001, 10110, 10011, 01110, 01101\} \text{ (weight 3 only)},$$
$$\{11110, 11101, 11011, 10111, 01111\} \text{ (weight 4 only)},$$
$$\{11111\} \text{ (weight 5)}. \quad (9)$$

Other choices can be made regarding which generations to merge to obtain a 3-write code from the $[5, 5 : 5, 3, 2, 1, 1]^{\mathrm{sync}}$ code, but lower WOM-rates are obtained.

IV. A Construction for Synchronous WOM Codes of Higher t

We propose a construction to obtain synchronous codes for higher values of t by concatenating n' instances of a synchronous code of length n, and using a second synchronous code of length n' to decide, at each generation, which of the n' instances of the first code are going to be modified.

Theorem 1: Let C be an $[n, t : M_1, \ldots, M_t]$ synchronous code of WOM-rate R, and C' an $[n', t' : M'_1, \ldots, M'_{t'}]$ synchronous code of WOM-rate R'. Then there exists an $[nn', tt' : M_1 M'_1, \ldots, M_1 M'_{t'}, \ldots, M_t M'_1, \ldots, M_t M'_{t'}]$ synchronous code C_1 of WOM-rate $R_1 = \frac{t'}{n'} R + \frac{t}{n} R'$.

A formal proof is omitted here, and we only give the idea of how the construction works. We first consider only the case where $n' = t'$ (in which C' writes exactly one wit at each generation). The key idea is that the nn' wits of C_1 are divided into n' blocks of n wits, and the tt' generations are divided into t stages of t' generations. In the first stage, during each of the first t' generations we use the encoding function of the first generation of C to write in exactly one empty block of n cells. C' tells us which of the n' blocks is going to be written: it writes exactly one wit among n' per generation, which we map to one block among the n' blocks at each generation. After this first stage, each block contains a codeword of the first generation of C. During the second stage, we use the encoding function of the second generation of C, and C' once again points to the block that will be written. This process is repeated for all t stages. Thus, during the l-th generation of the p-th stage ($1 \leq p \leq t$, $1 \leq l \leq t'$), we pick messages in $\{1, \ldots, M_p \times M'_l\}$, and each message m_1 can be mapped to a pair of messages $(m, m') \in \{1, \ldots, M_p\} \times \{1, \ldots, M'_l\}$.

From the decoder perspective, at all times, either every block has codewords of the same generation p of C, or there are blocks at generation p and blocks at generation $p - 1$. Because C is synchronous, the decoder has knowledge of the value of p (the current stage). Let $\mathbf{c}' = (c'_1, \ldots, c'_{n'})$ where $c'_k = 0$ if the k-th block is at generation $p - 1$ and $c'_k = 1$ if the k-th block is at generation p. The decoder knows $\mathbf{c}'$ and therefore m'. However, it does not have knowledge of which block of wits was written last. Therefore, we do not directly encode m in the block that is written: instead, we encode a message m_0 such that by decoding every block of wits at generation p using C, and then taking the modulo M_p sum of the decoded messages in $\{1, \ldots, M_p\}$, we recover m. m_0 is (modulo M_p) m minus the sum of the decoded messages of all the blocks at generation p (the encoder must therefore be able to decode C).

Now, if $n' \neq t'$, C' may write several wits during some generations. When this happens, the encoder of C_1 writes in every block pointed at by C', once again so that the modulo M_p sum of the decoded messages is m. Here, there are more degrees of freedom than in the case $n' = t'$ (in which m_0 was fully determined). A simple way to deterministically choose the values that we encode in each of the blocks that will be written is to encode M_p in each block but the last, and then

t	Rate of non-dec. code from [4]	Rate of dec. with no data	With data Sync. code	Rate
4	1.8564	1.7694	$[3, 3 : 3, 1, 1]$	1.7941
			$[5, 3 : 5, 3, 6]$	1.8128
5	1.9664	1.8435	$[4, 4 : 4, 3, 1, 1]$	1.8995
6	2.1297	1.9633	$[5, 5 : 5, 3, 2, 1, 1]$	2.0400
7	2.1697	1.9663	$[6, 6 : 6, 5, 3, 1, 1, 1]$	2.0677
			$[8, 6 : 8, 4, 6, 3, 4, 2]$	2.0886

t	Rate of non-dec. code from [4]	Rate of dec. with no data	With data Sync. code	Rate
4	1.8564	1.8346	$[3, 3 : 3, 1, 1]$	1.8408
			$[5, 3 : 5, 3, 6]$	1.8455
5	1.9664	1.9358	$[4, 4 : 4, 3, 1, 1]$	1.9498
6	2.1297	2.0881	$[5, 5 : 5, 3, 2, 1, 1]$	2.1073
7	2.1697	2.1188	$[6, 6 : 6, 5, 3, 1, 1, 1]$	2.1442
			$[8, 6 : 8, 4, 6, 3, 4, 2]$	2.1494

encode the same m_0 as in the case $n' = t'$ (since adding M_p has no effect modulo M_p).

Let us now establish the WOM-rate R_1 of C_1.

$$
\begin{aligned}
R_1 &= \frac{\sum_{p=1}^{t} \sum_{l=1}^{t'} \log(M_p M'_l)}{nn'} \\
&= \frac{1}{nn'} \left(\log \left(\prod_{p=1}^{t} M_p^{t'} \right) + \log \left(\prod_{l=1}^{t'} (M'_l)^t \right) \right) \quad (10) \\
&= \frac{1}{nn'} \left(t' \cdot nR + t \cdot n'R' \right) = \frac{t'}{n'} R + \frac{t}{n} R'.
\end{aligned}
$$

A. Example

Let C be the $[4, 3 : 4, 3, 2]^{\text{sync}}$ code defined by:

	1	2	3	4
$\mathcal{D}_1^{-1}$	$\{0001\}$	$\{0010\}$	$\{0100\}$	$\{1000\}$
$\mathcal{D}_2^{-1}$	$\{1100, 0011\}$	$\{1010, 0101\}$	$\{1001, 0110\}$	$-$
$\mathcal{D}_3^{-1}$	$\{0111, 1011, 1101, 1110\}$	$\{1111\}$	$-$	$-$

Let C' be the $[2, 2 : 2, 1]^{\text{sync}}$ code defined by:

	1	2
$(\mathcal{D}'_1)^{-1}$	$\{01\}$	$\{10\}$
$(\mathcal{D}'_2)^{-1}$	$\{11\}$	$-$

The code C_1 obtained with the construction is a $[8, 6 : 8, 4, 6, 3, 4, 2]^{\text{sync}}$ code. Consider that the eight cells are in state $(\mathbf{c}_1, \mathbf{c}_2) = (1100, 0010)$. Let us first consider the decoding of the message. The generation in C of the first block $\mathbf{c}_1$ is 2, and that of the second block $\mathbf{c}_2$ is 1, thus $p = 2$ (the highest of the two) and $\mathbf{c}' = (10)$. The fact that C' is synchronous guarantees that only one encoding function of C' has $\mathbf{c}'$ in its range: here, it is the encoding function for $l = 1$. Thus, we are at the first generation ($l = 1$) of the second stage ($p = 2$), so the overall generation is $i = (p-1)t' + l = (2-1) \times 2 + 1 = 3$.

We have $m' = \mathcal{D}'_1(10) = 2$. m is the modulo M_p sum of $\mathcal{D}_p(\mathbf{c}_k)$ for all indices k of a block at generation p of C. Here, there is only one block at generation $p = 2$ for C: block $\mathbf{c}_1 = (1100)$, therefore $m = \mathcal{D}_2(\mathbf{c}_1) \pmod 3 = 1$. The original message pair was therefore $(1, 2)$. This can be mapped to $m_1 \in \{1, \ldots, M_p \times M'_l\}$ by $m_1 = (m-1)M'_l + m'$, for instance, which gives $m_1 = 0 \times 2 + 2 = 2$.

Let us now encode a new message $m_1 = 2 \in \{1, 2, 3\}$ for generation 4. Our new m and m' are 2 and 1, respectively, so that $(m-1)M'_l + m' = (2-1) \times 1 + 1 = 2$. $\mathbf{c}' = (10)$ will become $\mathbf{c}' = (11)$ because $\mathcal{E}'_2(1, 10) = (11)$. Therefore, the second block is going to be written (because the second wit of $\mathbf{c}'$ changes). We first decode all the blocks already at generation $p = 2$: here, we only have one block at generation $p = 2$, and $\mathcal{D}_2(\mathbf{c}_1) = \mathcal{D}_2(1100) = 1$. We therefore encode in the second block $\mathbf{c}_2$ a message m_0 such that $1 + m_0 = m \pmod{M_p}$, where $M_p = M_2 = 3$ and $m = 2$. Thus, $m_0 = 1$. $\mathbf{c}_2$ is then replaced by $\mathcal{E}_p(1, 0010) = (0011)$. The state of the cells is $(1100, 0011)$ after this encoding phase.

B. Results

Let us denote by $F(C, C')$ the code obtained by applying the construction of Theorem 1 to C and C'. We can iterate the above construction by choosing C and C', and then defining $C_0 = C$ and for all $m > 0$, $C_m = F(C_{m-1}, C')$. This generates codes with even higher values of t, which have to be compared with a construction of synchronous codes from [1] (where $n = t$ is any power of two and the WOM-rate is $\log_2(t)/2$). Notice that the two constructions happen to match when we take as $C = C'$ the trivial $[2, 2 : 2, 1]^{\text{sync}}$ code.

First, we restrict ourselves to codes with $n = t$ (which are easier to compare) and we fix $C' = C$. The WOM-rate of the t_m-write code C_m after m iterations of the construction is

$$\mathcal{R}(C_m) = m\mathcal{R}(C) = \log_t(t_m)\mathcal{R}(C) = \frac{\mathcal{R}(C)}{\log_2(t)}\log_2(t_m).$$
(11)

Therefore, for codes with $n = t$, the higher $\frac{\mathcal{R}(C)}{\log_2(t)}$ is, the better this iterated construction works. The code that maximizes this ratio among those found by our computer search is the one with $n = t = 2$ (with $\frac{\mathcal{R}(C)}{\log_2(t)} = \frac{1}{2}$), making the codes from [1] the best in terms of asymptotic WOM-rate until codes for higher values of $n = t$ are found. For instance, Table I suggests that a $[8, 8 : 8, 7, 5, 5, 2, 2, 1, 1]^{\text{sync}}$ code could exist, with a ratio of 0.519 (and even better synchronous codes could exist even for $n = t = 8$, if we remove the added constraints from Section II). However, when the code length is not a power of two, our construction yields codes with lengths of the form $2^a 3^b 5^c$ by mixing different elementary codes of lengths 2, 3, and 5 as the C' codes, instead of always using the length-2 code. This is a much denser coverage of the potential values of t. Furthermore, if we consider codes with n slightly greater than t, we can reach higher WOM-rates at equal values of t. Consider, for instance, the code $F(C, C')$ with C the $[4, 3 : 4, 3, 2]^{\text{sync}}$ code and C' the $[2, 2 : 2, 1]^{\text{sync}}$ code. The construction then yields a $[8, 6 : 8, 4, 6, 3, 4, 2]^{\text{sync}}$ code of WOM-rate 1.512 (larger than $\log_2(t)/2$ both for $t = 6$ and $t = 8$). This is the example code of Section IV-A.

V. RESULTS

We compare our method of making WOM codes decodable with the method that adds $t - 1$ data-less cells. For this comparison, we consider two different target code lengths: $n = 64$ and $n = 256$. We then assume, for each value of n and for t between 4 and 7, that there is a code with WOM-rate equal to the best currently known WOM-rate for t-write codes (from [4]), and with length n minus the length of the synchronous code that we concatenate to it. We do not use the actual code lengths at which these state-of-the-art WOM-rates are reached because they are very large [5].

The results are reported in Tables II and III. The second column of each table reports the state-of-the-art WOM-rate of non-decodable codes, for each value of t. The third column shows the WOM-rate that would be obtained by appending $t - 1$ data-less cells to a code of length $n - t + 1$ and WOM-rate equal to the one reported in the second column. The last two columns show, for various synchronous codes, the WOM-rate that we obtain for the same target length. The $[3, 3 : 3, 1, 1]^{\text{sync}}$, $[4, 4 : 4, 3, 1, 1]^{\text{sync}}$, $[5, 5 : 5, 3, 2, 1, 1]^{\text{sync}}$, and $[6, 6 : 6, 5, 3, 1, 1, 1]^{\text{sync}}$ codes are from Section II, the $[5, 3 : 5, 3, 6]^{\text{sync}}$ code is from Section III, and the $[8, 6 : 8, 4, 6, 3, 4, 2]^{\text{sync}}$ code is from the construction of Section IV. Note that our technique yields higher WOM-rates compared to just appending a block of $t - 1$ cells with no information, for both target lengths. These WOM-rates are (to the best of our knowledge) also higher than the best WOM-rates for binary multiple-write codes (and hence better than the rates of any directly decodable code) known prior to [4], which justifies our approach.

VI. CONCLUSION

In this paper, we proposed short synchronous WOM codes as a basic tool to make non-decodable WOM codes decodable while preserving the WOM-rate as much as possible. We derived bounds for a simple family of short synchronous WOM codes, and constructed some synchronous WOM codes for small values of t. Furthermore, we proposed a construction method to build synchronous WOM codes for higher values of t obtained by concatenating shorter synchronous WOM codes.

REFERENCES

[1] R. L. Rivest and A. Shamir, "How to reuse a "write-once" memory," *Information and Control*, vol. 55, no. 1-3, pp. 1–19, Oct./Nov./Dec. 1982.

[2] R. Gabrys, E. Yaakobi, L. Dolecek, P. H. Siegel, A. Vardy, and J. K. Wolf, "Non-binary WOM-codes for multilevel flash memories," in *Proc. IEEE Inform. Theory Workshop*, Paraty, Brazil, Oct. 2011, pp. 40–44.

[3] E. Yaakobi, S. Kayser, P. H. Siegel, A. Vardy, and J. K. Wolf, "Efficient two-write WOM-codes," in *Proc. IEEE Inform. Theory Workshop*, Dublin, Ireland, Aug./Sep. 2010.

[4] S. Kayser, E. Yaakobi, P. H. Siegel, A. Vardy, and J. K. Wolf, "Multiple-write WOM-codes," in *Proc. 48th Annual Allerton Conf. Commun., Control, and Computing*, Monticello, IL, Oct. 2010, pp. 1062–1068.

[5] S. Kayser, private communication.