Coding for a Model of Random-Access Memories of a Computer and Write-Unidirectional Memories *

Vladimir B. Balakirsky
Member, IEEE

The author is currently with the
Fakultät für Mathematik, Universität Bielefeld,
Postfach 100131, D-33501 Bielefeld 1, Germany
on leave from the Data Security Association "Confident",
193060 St.-Petersburg, Russia
E-mail addresses :
vbal@mathematik.uni-bielefeld.de
vbal@vbal.spb.su

Abstract[1]**.** The OR-NOT operation $\mathbf{x} \vee \overline{\mathbf{y}}$ between two binary vectors $\mathbf{x}$ and $\mathbf{y}$ of the same length is viewed as an operation between the new and old content of a random-access memory used to increase the speed of the access to the memory. Coding procedures that allow us to recover $\mathbf{x}$ based on $\mathbf{x} \vee \overline{\mathbf{y}}$ are developed and connections of this model of the memory with the model of write-unidirectional memory are established.

Key words :
Random-Access Memory, Write-Unidirectional Memory, Coding, Decoding

---

*The work was supported by the SFB-343, Universität Bielefeld, Germany

[1] *SFB343 "Diskrete Strukturen in der Mathematik"*, Preprint 97-058, Bielefeld 1997. Submitted to the *Journal of Computer and System Sciences*, May 1997.

1 Introduction and Statement of the Problem

The study of different models of the memory from the point of view of information theory was initiated by Kuznetsov and Tsybakov [1], who introduced coding problems for the memory with defective cells (some cells always output fixed bits regardless of information that was stored). Rivest and Shamir [2], Wolf, Wyner, Ziv, and Körner [3], and the other authors analyzed a model that was called "Write-Once Memory" - WOM : if $\mathbf{x}_t \in \{0,1\}^n$ is a binary vector to be stored in the WOM containing a binary vector $\mathbf{y}_{t-1} \in \{0,1\}^n$, then

$$\begin{aligned} \mathbf{y}_0 &= 0^n, \\ \mathbf{y}_t &= \mathbf{x}_t \vee \mathbf{y}_{t-1}, \quad t = 1, 2, \ldots, \end{aligned} \tag{1}$$

where 0^n denotes the all-zero vector of length n. Another model, the so called "Write-Unidirectional Memory" - WUM, was introduced by Borden [4] and Willems and Vinck [5] :

$$\begin{aligned} \mathbf{y}_0 &= 0^n, \\ \mathbf{y}_t &= \begin{cases} \mathbf{x}_t \vee \mathbf{y}_{t-1}, & \text{if } t \text{ is odd,} \\ \mathbf{x}_t \wedge \mathbf{y}_{t-1}, & \text{if } t \text{ is even,} \end{cases} \quad t = 1, 2, \ldots \end{aligned} \tag{2}$$

Transformations (2) are unidirectional in the sense that one can replace either some 0's in the memory with 1's when t is odd, or some 1's with 0's when t is even. A more general model, where updating the memory should be realized under cost constraints ("Write-Efficient Memory" - WEM), was introduced by Ahlswede and Zhang [6]. There also exists a model of "Write-Isolated Memory" - WIM, where no change of two consecutive positions of $\mathbf{y}_{t-1}$ is allowed, introduced by Cohen and Zemor [7].

The list of models mentioned above is not complete [8], but all of them are oriented to an effective storage of the data that can be viewed as some operation with external memory of a computer. In the considerations below we are mainly concerned with internal or random-access memory where fast data exchange is important. Nevertheless, there are close connections between the model we will present and the WUM model.

The random-access memories are usually composed using the elements OR-NOT (NOR) and AND-NOT (NAND) : if $x, y \in \{0,1\}$ are inputs of these elements, then the output is defined as $\overline{x \vee y}$ and $\overline{x \wedge y}$, respectively. The common solutions of the data exchanging problem can be viewed as storing data in two steps : clean the memory and then write data; note that the cleaning procedure can be replaced with an equivalent one like selection of the input that should be used for entering a bit in a D flip-flop [9]. The models (1) and (2) do not use the cleaning, and with reference to possible applications to random-access memories they could be called the OR and OR/AND model. In the OR model the memory is used finitely many times since we may put a 1 into some cell only once. In the OR/AND model synchronization between the writings is required since the operation applied depends on t. We introduce a model of the memory that is specified by a time-invariant operation and can be used infinitely long,

$$
\begin{aligned}
\mathbf{y}_0 &= 1^n, \\
\mathbf{y}_t &= \mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1}, \quad t = 1, 2, \ldots,
\end{aligned}
\tag{3}
$$

where 1^n denotes the all-one vector of length n, and call it the ORN model. Note that the codes constructed for this model are implemented using the inverted versions of all codewords and the NAND elements because

$$
\mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1} = \overline{\overline{\mathbf{x}}_t \wedge \mathbf{y}_{t-1}}. \tag{4}
$$

We will analyze uniquely decodable coding for the ORN memory (Fig.1). Let M and n be given integers. At time $t \geq 1$, the message $m_t \in \{1, \ldots, M\}$ is encoded by a binary vector $\mathbf{x}_t \in \{0,1\}^n$, stored as a vector $\mathbf{y}_t \in \{0,1\}^n$ which is constructed in accordance with (3), and recovered by the decoder. Delay in the scheme is not allowed, i.e., the encoder forms $\mathbf{x}_t$ as soon as he gets m_t, and the decoder recovers m_t as soon as he gets $\mathbf{y}_t$. Coding for the ORN memory can be presented as assignment of a deterministic encoding function f taking values in $\{0,1\}^n$ and a deterministic decoding function φ taking values in $\{1, \ldots, M\}$. We follow the approaches developed for the other models of memory and say that $\mathbf{x}_t$ may depend on the previous messages only through the current content of the memory $\mathbf{y}_{t-1}$, and E_+ denotes the case when f has the arguments m_t and $\mathbf{y}_{t-1}$. Similarly, a possible help to the decoder can be given via the vector $\mathbf{y}_{t-1}$, and D_+ denotes the case when φ has the arguments $\mathbf{y}_t$ and $\mathbf{y}_{t-1}$. The cases when $\mathbf{y}_{t-1}$ is not available to the encoder and decoder are denoted by E_- and D_-, respectively. Thus, the complete coding situation is specified by a pair $(E_{(\cdot)}, D_{(\cdot)})$, where $(\cdot) \in \{+, -\}$. Note however, that the case (E_-, D_-) is out of sense for the model of memory defined by (3): if $m_2 = m_1$, then the decoder receives $\mathbf{y}_2 = 1^n$ for any encoding f. Therefore, we introduce the case $(E_-, D_-)'$ adding the restriction : $m_t \neq m_{t-1}$ for all $t > 1$. The list of four cases is given in Table 1. In the further considerations we will abbreviate a uniquely decodable code for the ORN memory in the case $(E_{(\cdot)}, D_{(\cdot)})$ as an $\mathrm{ORN}_{(\cdot,\cdot)}$ code [2] and sometimes unify the $'+'$ and $'-'$ cases writing $f(m_t, \times)$, where the $'\times'$ sign designates a possible dependence on $\mathbf{y}_{t-1}$ (the encoding $f(m_t, \times)$ should be understood as $f(m_t)$ in the case E_-).

The ORN memory has the following important property : any sequence of the contents of the j-th cell, $y_0^{(j)}, y_1^{(j)}, y_2^{(j)}, \ldots,$ does not contain 0's at two neighboring positions (if $y_{t-1}^{(j)} = 0$, then $x_t^{(j)} \vee \overline{y_{t-1}^{(j)}} = 1$), i.e.,

$$y_{t-1}^{(j)} \vee y_t^{(j)} = 1, \tag{5}$$

for all $j = 1, \ldots, n$ and $t = 1, 2, \ldots$ It means that if the j-th cell contains 0 at time $t-1$, then the j-th position can be viewed as a closed one at time t : the decoder receives 1 regardless of the transmitted bit. However, at time $t+1$, this cell will contain 1, which means that the decoder will receive the transmitted bit. If this bit is 0, then the position becomes closed again. The storing transformation is a kind of information transmission over a channel where the channel at time t is created by the previously transmitted codewords, mainly the last codeword. Note also that transmission of the vector 1^n can be considered as cleaning the memory because, in this case, any of 2^n binary vectors can be noiselessly transmitted at the next time instant.

The ORN and WUM models are closely related to each other. Suppose that the encoding function of an ORN code is specified as $f = \{f(m, \times)\}$. Then we may construct the WUM code for the same case $(E_{(\cdot)}, D_{(\cdot)})$ defined by the initial content of the memory $\mathbf{y}'_0 = 0^n$ and encoding functions $f_e = \{f_e(m, \times)\}$ for even time instants and $f_o = \{f_o(m, \times)\}$ for odd time instants

[2] We use $'\cdot'$ in the context that the value can be independently assigned as $+$ and $-$ at any place where the $'\cdot'$ sign is written.

such that

$$f_e(m,\times)=f(m,\times),\quad f_o(m,\times)=\overline{f(m,\overline{\times})},\quad \text{for all } m\in\{1,\dots,M\}. \tag{6}$$

Equations (6) for f_o mean that to encode the message m at odd time instant the WUM encoder inverts $\mathbf{y}_{t-1}$ and takes the inverted codeword of the ORN code corresponding to the pair $(m,\overline{\mathbf{y}}_{t-1})$ (we consider the case E_+ as an example). Note that the WUM codes satisfying

$$f_o(m,\times)=\overline{f_e(m,\overline{\times})}$$

for all $m\in\{1,\dots,M\}$ are known as alternating codes [10]. If some sequence of messages is encoded using the ORN code by binary vectors $\mathbf{x}_1,\mathbf{x}_2,\dots$ and stored as a sequence $\mathbf{y}_0=1^n,\mathbf{y}_1,\mathbf{y}_2,\dots$ in the ORN memory, then the same sequence of messages will be encoded by binary vectors $\mathbf{x}'_1,\mathbf{x}'_2,\dots$ using the WUM code and stored as a sequence $\mathbf{y}'_0=0^n,\mathbf{y}'_1,\mathbf{y}'_2,\dots$ in the WUM, where

$$(\mathbf{x}'_t,\mathbf{y}'_t)\quad=\quad\begin{cases}(\mathbf{x}_t,\mathbf{y}_t), & \text{if } t \text{ is odd},\\ (\overline{\mathbf{x}}_t,\overline{\mathbf{y}}_t), & \text{if } t \text{ is even},\end{cases}\quad t=1,2,\dots \tag{7}$$

This statement can be easily checked by induction,

$$\begin{aligned} t=0 \quad&\Longrightarrow\quad \mathbf{y}'_0=\overline{\mathbf{y}}_0, \\ t \text{ is odd},\ \mathbf{y}'_{t-1}=\overline{\mathbf{y}}_{t-1}\quad&\Longrightarrow\quad \mathbf{y}'_t=\mathbf{x}'_t\vee\mathbf{y}'_{t-1}=\mathbf{x}_t\vee\overline{\mathbf{y}}_{t-1}=\mathbf{y}_t,\\ t \text{ is even},\ \mathbf{y}'_{t-1}=\mathbf{y}_{t-1}\quad&\Longrightarrow\quad \mathbf{y}'_t=\mathbf{x}'_t\wedge\mathbf{y}'_{t-1}=\overline{\mathbf{x}}_t\wedge\mathbf{y}_{t-1}=\overline{\mathbf{x}_t\vee\overline{\mathbf{y}}_{t-1}}=\overline{\mathbf{y}}_t,\end{aligned} \tag{8}$$

where identity (4) was used. Transformation in an opposite direction (constructing the ORN code based on an alternating WUM code) is also possible: we define the ORN code by the initial content of the memory $\mathbf{y}_0=1^n$ and the encoding function

$$f(m,\times)=f_e(m,\times),\quad \text{for all } m\in\{1,\dots,M\}.$$

Then (7) can be represented in the following way : if a sequence of messages is encoded as $\mathbf{x}'_1,\mathbf{x}'_2,\dots$ and stored as a sequence $\mathbf{y}'_0=0^n,\mathbf{y}'_1,\mathbf{y}'_2,\dots$ in the WUM, then the same sequence of messages will be encoded as $\mathbf{x}_1,\mathbf{x}_2,\dots$ and stored as a sequence $\mathbf{y}_0=1^n,\mathbf{y}_1,\mathbf{y}_2,\dots$ in the ORN memory, where

$$(\mathbf{x}_t,\mathbf{y}_t)\quad=\quad\begin{cases}(\mathbf{x}'_t,\mathbf{y}'_t), & \text{if } t \text{ is odd},\\ (\overline{\mathbf{x}'_t},\overline{\mathbf{y}'_t}), & \text{if } t \text{ is even},\end{cases}\quad t=1,2,\dots$$

The proof is similar to (8),

$$\begin{array}{rcl} t = 0 & \Longrightarrow & \mathbf{y}_0 = \overline{\mathbf{y}'_0}, \\ t \text{ is odd}, \ \mathbf{y}_{t-1} = \overline{\mathbf{y}'_{t-1}} & \Longrightarrow & \mathbf{y}_t = \mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1} = \mathbf{x}'_t \vee \mathbf{y}'_{t-1} = \mathbf{y}'_t, \\ t \text{ is even}, \ \mathbf{y}_{t-1} = \mathbf{y}'_{t-1} & \Longrightarrow & \mathbf{y}_t = \mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1} = \mathbf{x}_t \vee \overline{\mathbf{y}'_{t-1}} = \overline{\overline{\mathbf{x}}_t \wedge \mathbf{y}'_{t-1}} = \overline{\mathbf{y}'_t}. \end{array}$$

The fact that the encoder always knows whether t is even or odd is very important for the WUM model, i.e., the encoder is not completely ignorant about the situation with the memory in the case E_-. This peculiarity may essentially affect both the applications and theoretical studies [11]. In particular, the code presented by Borden [4] for the case (E_-, D_-) is as follows. Suppose that $M = 2^n$ and denote a binary representation of the integer $m-1$ of length n by $b(m-1)$, where $m \in \{1, \ldots, 2^n\}$. At odd time instants t, the encoder sends $(1^n, b(m_t - 1))$ and, at even time instants t, he sends $(b(m_t - 1), 0^n)$, i.e., the length of the codewords is $2n$. This transmission scheme exactly matches the situation we discussed earlier : the storing process for the ORN memory can be organized as consisting of two steps, - first we clean the memory by sending 1^n and then transmit any binary vector of length n. Instead of doubling the time we can double the memory simultaneously sending a codeword to one of the halves and cleaning the other half. At the next time instant we exchange the roles of the halves. This situation differs from the case E_- that we defined for the ORN memory, i.e., we want to construct codes when the encoder is completely ignorant and does not know which part of the memory should be used for data transmission.

Suppose that the messages to be delivered to the decoder at each time instant are binary vectors of length K and that we are given an $\mathrm{ORN}_{(\cdot,\cdot)}$ code of length n for M messages.[3] In the cases (E_+, D_+), (E_+, D_-), and (E_-, D_+) the encoder represents the t-th input message by a vector $(m_t^{(1)}, \ldots, m_t^{(N)})$, where $m_t^{(1)}, \ldots, m_t^{(N)} \in \{0, \ldots, M-1\}$, $N = \lceil K/\log M \rceil$, and $\lceil z \rceil$ denotes the smallest integer which is not less than z. Afterwards the sequence $(f(m_t^{(1)} + 1, \times), \ldots, f(m_t^{(N)} + 1, \times))$ of length $nN = n\lceil K/\log M \rceil$ is transmitted to the memory. In the case $(E_-, D_-)'$ the encoder represents the t-th message by a vector $(m_t^{(1)}, \ldots, m_t^{(N)})$, where $m_t^{(1)}, \ldots, m_t^{(N)} \in \{0, \ldots, M-2\}$, $N = \lceil K/\log(M-1) \rceil$, and transmits the sequence $(f(\tilde{m}_t^{(1)} + 1), \ldots, f(\tilde{m}_t^{(N)} + 1))$

[3] Hereafter, $\log z$ is understood as $\log_2 z$ for all $z > 0$. We also set $0 \log 0 = 0$.

of length $nN = n\lceil K/\log(M-1)\rceil$, where $\tilde{m}_t^{(i)} = m_t^{(i)}$ if $m_t^{(i)} < m_{t-1}^{(i)}$ and $\tilde{m}_t^{(i)} = m_t^{(i)} + 1$ if $m_t^{(i)} \geq m_{t-1}^{(i)}$, $i = 1, \ldots, N$. In both cases the decoder can recover $(m_t^{(1)}, \ldots, m_t^{(N)})$ and reconstruct the original message. Hence, an efficiency of the ORN code can be measured by its rate

$$R = \begin{cases} (\log M)/n, & \text{for cases } \ (E_+, D_+), (E_+, D_-), (E_-, D_+), \\ (\log(M-1))/n, & \text{for case } \ (E_-, D_-)', \end{cases} \tag{9}$$

and we are interested in codes that can be used for storing a maximal possible number of messages when the length is fixed.

The paper is organized as follows. In Section 2 we prove necessary and sufficient conditions for the ORN codes to be uniquely decodable, present specific codes, and illustrate coding schemes. These codes are mainly constructed by translation of the known codes for the WUM model. Section 3 is devoted to asymptotic characteristics of the ORN codes : we are looking for a potential rate that can be attained for the ORN model of the memory since the answer is useful for understanding how far is a specific code from being an optimal one. The proofs for the cases (E_+, D_+) and (E_+, D_-) use the ideas developed for the WUM codes [3], [4], [10], while the analysis of the case (E_-, D_+) is new.

2 Specific ORN Codes and Their Decoding

The transmitted codeword of an ORN code serves both for delivering a message to the decoder and creating a "storage channel" for the next messages. The encoding function $f = \{f(m, \times)\}$ generates a set of available contents of the memory $\mathcal{Y}^n \subseteq \{0,1\}^n$ that can be constructed iteratively :

a) $\mathcal{Y}_0^n := \{1^n\}$ and $t := 1$;

b) $\mathcal{Y}_t^n := \bigcup_{m=1}^M \bigcup_{\mathbf{y} \in \mathcal{Y}_{t-1}^n} \{f(m, \times) \vee \overline{\mathbf{y}}\}$;

c) if $\mathcal{Y}_t^n \neq \mathcal{Y}_{t-1}^n$, increase t by 1 and repeat the previous step;

d) $\mathcal{Y}^n := \mathcal{Y}_t^n$.

The requirement that the ORN code is uniquely decodable means that each message must be recovered by the decoder when the corresponding codeword is stored in the memory containing any vector $\mathbf{y} \in \mathcal{Y}^n$.

For specific codes under considerations in this section a simpler scheme of generating $\mathcal{Y}^n$ is developed : first we fix an integer w and introduce the set of binary vectors of length n having at least w ones,

$$\mathcal{B}_w^n = \{ \mathbf{y} \in \{0,1\}^n : \; w_H(\mathbf{y}) \geq w \}, \tag{10}$$

where w_H denotes the Hamming weight. If all codewords of an ORN code belong to $\mathcal{B}_w^n$, then the result of the OR operation of a codeword with any binary vector also belongs to $\mathcal{B}_w^n$. Hence, $\mathcal{Y}^n \subseteq \mathcal{B}_w^n$ and we want to uniquely decode the messages for all $\mathbf{y}_{t-1} \in \mathcal{B}_w^n$. Nevertheless, the necessary and sufficient conditions for the ORN codes to be uniquely decodable ("iff" stands for "if and only if"), are given below for a general case.

Lemma

(1) $\{f(m,\mathbf{y})\}$ *is an* $ORN_{(+,+)}$ *code iff* M *vectors*

$$f(1,\mathbf{y}) \vee \overline{\mathbf{y}}, \ldots, f(M,\mathbf{y}) \vee \overline{\mathbf{y}}$$

are distinct for any $\mathbf{y} \in \mathcal{Y}^n$.

(2) $\{f(m,\mathbf{y})\}$ *is an* $ORN_{(+,-)}$ *code iff* M *sets*

$$\bigcup_{\mathbf{y} \in \mathcal{Y}^n} \{f(1,\mathbf{y}) \vee \overline{\mathbf{y}}\}, \ldots, \bigcup_{\mathbf{y} \in \mathcal{Y}^n} \{f(M,\mathbf{y}) \vee \overline{\mathbf{y}}\}$$

are pairwise disjoint.

(3) $\{f(m)\} = \{\mathbf{x}(m)\}$ *is an* $ORN_{(-,+)}$ *code iff* M *vectors*

$$\mathbf{x}(1) \vee \overline{\mathbf{x}(i)}, \ldots, \mathbf{x}(M) \vee \overline{\mathbf{x}(i)}$$

are distinct for any $i = 1, \ldots, M$, *or, equivalently,*

$$\left| \bigcup_{m=1}^{M} \{\mathbf{x}(m) \wedge \mathbf{x}(i)\} \right| = M, \quad i = 1, \ldots, M. \tag{11}$$

(4) $\{f(m)\} = \{\mathbf{x}(m)\}$ *is an* $ORN'_{(-,-)}$ *code iff* M *sets*

$$\bigcup_{i=1}^{M}\{\mathbf{x}(1) \vee \overline{\mathbf{x}(i)}\}, \ldots, \bigcup_{i=1}^{M}\{\mathbf{x}(M) \vee \overline{\mathbf{x}(i)}\}$$

are pairwise disjoint.

Proof

The statements *(1)–(2)* directly follow from the requirement that the $\mathrm{ORN}_{(+,\cdot)}$ codes are uniquely decodable.

The necessary condition of *(3)* follows from the fact that the decoder of an $\mathrm{ORN}_{(-,+)}$ code can distinguish between the sequences of messages $i, m, \ldots$ and $i, m', \ldots$ for all $i, m, m' \in \{1, \ldots, M\}$, $m' \neq m$; the statement concerning the equivalence follows from the observation

$$x \vee \overline{y} = x' \vee \overline{y} \iff x \wedge y = x' \wedge y, \text{ for all } x, x', y \in \{0,1\}.$$

To prove the sufficient condition of *(3)* note that any given $\mathbf{y} \in \mathcal{Y}^n$ is obtained as a result of the OR operation of some codeword $\mathbf{x}(i)$ with another $\mathbf{y}' \in \mathcal{Y}^n$. Therefore, if the vectors $\mathbf{x}(1) \vee \overline{\mathbf{x}(i)}, \ldots, \mathbf{x}(M) \vee \overline{\mathbf{x}(i)}$ are distinct, then the vectors $\mathbf{x}(1) \vee \overline{\mathbf{y}}, \ldots, \mathbf{x}(M) \vee \overline{\mathbf{y}}$ are also distinct. This fact allows the decoder to make a unique decision about m.

The statement *(4)* can be proved similarly. To prove the necessary condition we refer to the sequences $i, m, \ldots$ and $i', m', \ldots$ for all $i, i', m, m' \in \{1, \ldots, M\}$ such that $m' \neq m$. To prove the sufficient condition we consider $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}^n$ and assign $i, i' \in \{1, \ldots, M\}$ in such a way that $\mathbf{y} \geq \mathbf{x}(i)$ and $\mathbf{y}' \geq \mathbf{x}(i')$.

□

Let us consider constructions of specific ORN codes. For any content of the memory $\mathbf{y}_t = (y_{t1}, \ldots, y_{tn})$, we denote

$$\mathcal{O}_t = \{\, j \in \{1, \ldots, n\} : \; y_{tj} = 0 \,\}.$$

At the next time instant this set becomes $\mathcal{O}_{t-1}$ and includes the positions where the decoder receives 1's regardless of transmitted bits. If the decoder is informed (the case D_+), then he punctures corresponding positions and we mark these positions by the $'*'$ symbol in the included tables illustrating the coding schemes.

We will use the binary entropy function

$$h(z) = -z \log z - (1-z)\log(1-z) \tag{12}$$

defined for all $z \in [0,1]$ and the well-known approximation of the sum of binomial coefficients (see, for example [12], Chapter 7.1) : if $w \ge n/2$ and proportional to n, then

$$\frac{1}{n}\log \sum_{i=0}^{n-w} \binom{w}{i} = \frac{w}{n} \cdot h\left(\frac{n-w}{w}\right) - \delta_n, \tag{13}$$

where $\delta_n \to 0$, as $n \to \infty$.

The case (E_+, D_+).

Let the length of the codewords be divisible by 3 and let $\mathbf{c}(1), \ldots, \mathbf{c}(M)$, where

$$M = \sum_{i=0}^{n/3} \binom{2n/3}{i} = \frac{1}{2} 2^{2n/3} + \frac{1}{2}\binom{2n/3}{n/3}, \tag{14}$$

denote the list of binary vectors of length $2n/3$ containing at most $n/3$ zeroes. This list can be used as an auxiliary code for the $\mathrm{ORN}_{(+,+)}$ code of length n with $\mathcal{Y}^n = \mathcal{B}^n_{2n/3}$. First the encoder and decoder use some regular algorithm and extend $\mathcal{O}_{t-1}$ to the set $\mathcal{O}'_{t-1}$ consisting of $n/3$ elements; for example, $n/3 - |\mathcal{O}_{t-1}|$ minimal numbers of $\{1, \ldots, n\} \backslash \mathcal{O}_{t-1}$ can be added to $\mathcal{O}_{t-1}$. Then the encoder transmits the components of the vector $\mathbf{c}(m_t)$ at positions $j \notin \mathcal{O}'_{t-1}$ and 1's at positions $j \in \mathcal{O}'_{t-1}$. This codeword will be noiselessly delivered to the decoder who can puncture the positions belonging to $\mathcal{O}'_{t-1}$, read $\mathbf{c}(m_t)$, and decode m_t. The codeword has at most $n/3$ zeroes; hence, $|\mathcal{O}_t| \le n/3$. This fact makes the use of the same procedure possible at the next time instant. Tables 2,3 illustrate the coding scheme.

The procedure described above exchanges the role of zeroes and ones of the codewords and reminds us of the well-known algorithm of dividing the interval when we are looking for the maximal value of a convex up function. This analogy really makes sense, and the Golden proportion comes to the answer when we are interested in a maximal possible value of the code rate

R of an $\mathrm{ORN}_{(+,+)}$ code. Let us fix $\gamma \in [1/2, 1]$ in such a way that $n\gamma$ is an integer and denote by $\mathbf{c}(1), \dots, \mathbf{c}(M)$, where

$$M = \sum_{i=0}^{n(1-\gamma)} \binom{n\gamma}{i},$$

the list of binary vectors of length $n\gamma$ having at most $n(1-\gamma)$ zeroes. The considerations above can be repeated replacing $n/3$ and $2n/3$ with $n(1-\gamma)$ and $n\gamma$, respectively. As a result, we construct an $\mathrm{ORN}_{(+,+)}$ code of length n for M messages and using (13) conclude that

$$R = \frac{\log M}{n} = \gamma h\left(\frac{1-\gamma}{\gamma}\right) - \delta_n.$$

However,

$$\gamma h\left(\frac{1-\gamma}{\gamma}\right) = -(1-\gamma)\log(1-\gamma) + \gamma\log\gamma - (2\gamma-1)\log(2\gamma-1)$$

is a convex up function of γ and taking the derivative we obtain the equation

$$\log(1-\gamma_0) + \log\gamma_0 - \log(2\gamma_0 - 1)^2 = 0$$

for the value of γ_0 where this function is maximized. The solution is the Fibonacci number $\gamma_0 = 1/2 + 1/(2\sqrt{5})$. Thus, assigning the value of γ close to γ_0, we get

$$R \to \left(\frac{1}{2} + \frac{1}{2\sqrt{5}}\right) \cdot h\left(\frac{\sqrt{5}-1}{\sqrt{5}+1}\right) = \log\frac{1+\sqrt{5}}{2}, \quad \text{as } n \to \infty.$$

In the next section we show that this result cannot be improved.

The case (E_+, D_-).

Let us fix w and reduce the sufficient condition *(2)* of the Lemma to the following one. Let $\mathcal{X}(1), \dots, \mathcal{X}(M)$ be a collection of sets of binary vectors of length n such that

(1) $\mathcal{X}(1), \dots, \mathcal{X}(M)$ are pairwise disjoint, i.e., any pair of sets does not contain common vectors;

(2) each vector of Hamming weight at least w is included into one of the sets;

(3) for each vector $\mathbf{y}$ of Hamming weight at least w and each $m \in \{1, \ldots, M\}$ there exists at least one vector $\mathbf{x} \in \mathcal{X}(m)$ that covers $\overline{\mathbf{y}}$, i.e., $x_j \geq \overline{y}_j$ for all $j = 1, \ldots, n$ and we may write $\mathbf{x} \geq \overline{\mathbf{y}}$.

Then the encoding and decoding defined in such a way that

$$\begin{aligned} f(m, \mathbf{y}) = \mathbf{x} \quad &\Longrightarrow \quad \mathbf{x} \in \mathcal{X}(m), \;\; \mathbf{x} \geq \overline{\mathbf{y}}, \\ \varphi(\mathbf{x} \vee \overline{\mathbf{y}}) = m \quad &\Longleftrightarrow \quad \mathbf{x} \vee \overline{\mathbf{y}} \in \mathcal{X}(m) \end{aligned}$$

specify an $\mathrm{ORN}_{(+,-)}$ code.

Let us consider the example given in Table 4. All binary vectors of length $n = 5$ and Hamming weight at least 3 are rows of the matrices $\mathbf{X}(1), \ldots, \mathbf{X}(6)$ and each of them enters only one matrix, i.e., properties (1) and (2) hold for the sets of rows of these matrices. Let us check that the property (3) also holds. Let $m = 1$. We consider the vectors $\mathbf{y} = (y_1, \ldots, y_5)$ containing at most 2 zeroes. If $y_3 = 1$, then $11011 \geq \overline{\mathbf{y}}$. If $y_3 = 0$ and either $y_1 = 0$ or $y_2 = 0$, then $11100 \geq \overline{\mathbf{y}}$. If $y_3 = 0$ and either $y_4 = 0$ or $y_5 = 0$, then $00111 \geq \overline{\mathbf{y}}$. Hence, the statement is true for the first matrix and for the matrices $\mathbf{X}(2), \ldots, \mathbf{X}(5)$ obtained by permutations of the columns of $\mathbf{X}(1)$. Note also that $\mathbf{X}(6)$ contains the all-one vector which covers every binary vector. An example of coding is presented in Table 5.

As we noticed in the previous section, it is possible to use specific $\mathrm{WUM}_{(+,-)}$ codes to construct $\mathrm{ORN}_{(+,-)}$ codes. One $\mathrm{WUM}_{(+,-)}$ code of length 11 was discovered by Simonyi [10], but only the first matrix that defines a partition of the space consisting of binary vectors of Hamming weight 8 and 9 was published. We give all matrices in Table 6. References to the other known $\mathrm{WUM}_{(+,-)}/\mathrm{ORN}_{(+,-)}$ codes are given in Table 7.

The case (E_-, D_+).

In this case the encoder does not adopt transmitted codewords to the content of the memory. Therefore, a code can be defined by a binary $M \times n$ matrix and if the m-th message should be stored, then the m-th row of the matrix is sent to the memory. An example of such a matrix for $n = M = 3$

is

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{15}$$

At time instant t the previous content of the memory, $\mathbf{y}_{t-1} = (y_{t-1,1}, \ldots, y_{t-1,n})$ is known to the decoder who punctures all positions j where $\mathbf{y}_{t-1}$ contains 0's (if $y_{t-1,j} = 0$ then $y_{t,j} = 1$ for any codeword $\mathbf{x}_t$). All other positions of $\mathbf{y}_t$ contain corresponding components of the transmitted codeword. The coding procedure is illustrated in Table 8.

The problem of constructing $\mathrm{ORN}_{(+,-)}$ codes coincides with the problem of constructing the so-called cancellative sets [15] : there is a set of integers $\{1, \ldots, n\}$ and we want to find M sets $A_1, \ldots, A_M \subseteq \{1, \ldots, n\}$ having the following property :

$$A_m \bigcap A_i = A_m \bigcap A_j \implies A_i = A_j, \text{ for all } i, j, m \in \{1, \ldots, M\}.$$

Note that for any sets $A, B \subseteq \{1, \ldots, n\}$,

$$A \bigcup B = \overline{\overline{A} \bigcap \overline{B}},$$

where the line above denotes the complement to the set. Thus, based on a cancellative family $A_1, \ldots, A_M$ we can construct an $\mathrm{ORN}_{(+,-)}$ code defining the m-th codeword $\mathbf{x}(m) = (x_1(m), \ldots, x_n(m))$ as follows :

$$x_j(m) = \begin{cases} 0, & \text{if } j \in A_m, \\ 1, & \text{if } j \notin A_m. \end{cases}$$

A construction found by Shearer [16] uses the code specified in (15) and improves the code rate as compared to $(\log 3)/3 \approx 0.528$. This construction can be described as follows. Fix an integer k and set $(M, n) = (k \cdot 3^{k-2}, 3k)$. Represent the transmitted message as a k-tuple $\mathbf{m} = (m^{(1)}, \ldots, m^{(k)})$ with the components satisfying the following restrictions : $m^{(1)}, \ldots, m^{(k)} \in \{\sqcup, 0, 1, 2\}$; there exists exactly one $j_0 \in \{1, \ldots, k\}$ such that $m^{(j_0)} = \sqcup$; the arithmetic sum of all components at positions $j \neq j_0$ is divisible by 3. The codeword $\mathbf{x}(m)$ is a concatenation of k 3-tuples $\mathbf{v}(m^{(1)}), \ldots, \mathbf{v}(m^{(k)})$, where

$$\mathbf{v}(m^{(l)}) = \begin{cases} 000, & \text{if } m^{(l)} = \sqcup, \\ 011, & \text{if } m^{(l)} = 0, \\ 101, & \text{if } m^{(l)} = 1, \\ 110, & \text{if } m^{(l)} = 2. \end{cases}$$

It is easy to see that the resulting code is uniquely decodable and the code rate is equal to

$$R_k = \frac{\log(k \cdot 3^{k-2})}{3k}.$$

If $k > 9$, then $R_k > (\log 3)/3$, and the maximum of the expression at the right hand side is attained for $k = 24$ that gives $R_{24} \approx 0.5480$. The $\mathrm{ORN}_{(-,+)}$ code constructed for $k = 4$ is given in Table 9.

The case $(E_-, D_-)'$.

In this case both the encoder and decoder are ignorant about the previous content of the memory. The best code we could find is based on the matrix (15) and its rate is equal to 1/3 (see 9). An example of the coding procedure is shown in Table 10.

Note that the problem of constructing $\mathrm{ORN}'_{(-,-)}$ codes coincides with the following problem of the set theory : given an integer n construct M families $A_1, \ldots, A_M \subseteq \{1, \ldots, n\}$ having the following property :

$$A_m \bigcap \overline{A}_i = A_{m'} \bigcap \overline{A}_{i'} \implies A_i = A_{i'}, \text{ for all } \begin{array}{l} m, i, m', i' \in \{1, \ldots, M\} : \\ m' \neq m, \ i' \neq i. \end{array}$$

3 Asymptotic Characteristics of the ORN Codes

3.1 Formulation of the results

Theorem *Let $M_{(\cdot,\cdot)}(n)$ denote the maximal number of messages for which there exists an $ORN_{(\cdot,\cdot)}$ code of length n and let*

$$R_{(\cdot,\cdot)} = \lim_{n\to\infty} \frac{\log M_{(\cdot,\cdot)}(n)}{n}.$$

Then

$$\begin{aligned} R_{(+,+)} = R_{(+,-)} &= \max_{0\le\gamma\le1} F_{(+)}(\gamma) = F_{(+)}(0.7236\ldots) \approx 0.6942, \qquad (16) \\ R_{(-,+)} &\le \max_{0\le\gamma\le1} F_{(-)}(\gamma) = F_{(-)}(0.7035\ldots) \approx 0.6169, \end{aligned}$$

where
$$F_{(+)}(\gamma) = \gamma h\left(\frac{1-\gamma}{\gamma}\right), \quad F_{(-)}(\gamma) = \gamma h(\gamma), \tag{17}$$
and the binary entropy function $h(z), z \in [0,1]$, *is defined in (12). Furthermore,*
$$0.7236\ldots = \frac{1}{2} + \frac{1}{2\sqrt{5}}$$
is the Fibonacci number,
$$F_{(+)}(0.7236\ldots) = \log\frac{1+\sqrt{5}}{2}$$
is the limit of the logarithm of the t*-th element of the Fibonacci sequence divided by* t *when* $t \to \infty$, *and* $\gamma = 0.7035\ldots$ *is the root of the equation*
$$-2\gamma\log\gamma + (2\gamma-1)\log(1-\gamma) = 0.$$

3.2 Proof of converse statements

Proposition 1 *$ORN_{(\cdot,\cdot)}$ codes of length n for M messages may exist only if*
$$M \le \inf_{t\ge 1} a_{t+1}^{n/t} = \left(\frac{1+\sqrt{5}}{2}\right)^n,$$
where
$$a_{t+1} = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{t+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{t+1}\right]$$
is the $(t+1)$*-st component of the Fibonacci sequence* $(a_0, a_1, \ldots) =$ *1,1,2,3,5,...,* *i.e.,* $a_i = a_{i-1} + a_{i-2}$ *for all* $i \ge 2$.

Proof Let us fix $t \ge 1$ and $j \in \{1, \ldots, n\}$. Denote the total number of binary sequences $(y_0^{(j)} = 1, y_1^{(j)}, \ldots, y_t^{(j)})$ satisfying (5) by K_t (obviously, this number does not depend on j). Then the total number of sequences that can be observed by the decoder is upper-bounded by K_t^n. The decoder recovers any of M^t sequences of t messages; thus $M^t \le K_t^n$ and we complete the proof since $K_t = a_{t+1}$ (see, for example [17], pp.157-158).

$\square$

Proposition 2 *If* $\{\mathbf{x}(m)\}$ *is an ORN*$_{(-,+)}$ *code such that all codewords have the Hamming weight* w, *then*

$$\log M \leq \begin{cases} w, & \text{if } w \leq n/2, \\ w \cdot h(w/n), & \text{if } w > n/2. \end{cases} \tag{18}$$

Corollary 2 *If* $\{\mathbf{x}(m)\}$ *is an ORN*$_{(-,+)}$ *code, then*

$$\frac{\log M}{n} \leq \max_{0 \leq \gamma \leq 1} F_{(-)}(\gamma) + \frac{\log(n+1)}{n},$$

where the function $F_{(-)}(\gamma)$ *is defined in (17).*

Proof If $w \leq n/2$, then (18) directly follows from (11) : there are at most 2^w distinct vectors obtained as a result of the AND operation of some binary vector with a fixed vector of the Hamming weight w. Therefore, we restrict our attention to the case $w > n/2$. Let us construct an $M \times n$ matrix consisting of the codewords of an ORN$_{(-,+)}$ code, denote the frequency of 1's in the j-th column of the matrix by

$$\gamma_j = \frac{1}{M} \sum_{m=1}^{M} x_j(m) \tag{19}$$

and introduce probability distributions

$$(\gamma_j(0), \gamma_j(1)) = (1 - \gamma_j, \gamma_j), \quad j = 1, \ldots, n. \tag{20}$$

For any set $\mathcal{J} \subseteq \{1, \ldots, n\}$, let us define $\mathcal{X}(\mathcal{J})$ as a set consisting of the n-tuples whose components are 0's and 1's at positions included into $\mathcal{J}$ and special $'*'$ symbols at the other positions,

$$\mathcal{X}(\mathcal{J}) = \{\ \mathbf{x} = (x_1, \ldots, x_n) : \ x_j \in \{0,1\}, j \in \mathcal{J};\ x_j = *, j \notin \mathcal{J}\ \}.$$

Note that

$$\left\{ P(\mathbf{x}) = \prod_{j \in \mathcal{J}} \gamma_j(x_j),\ \mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}(\mathcal{J}) \right\} \tag{21}$$

is the probability distribution on $\mathcal{X}(\mathcal{J})$ and

$$\sum_{\mathbf{x}\in\mathcal{X}(\mathcal{J})} P(\mathbf{x}) = 1. \tag{22}$$

Let us fix $i \in \{1, \ldots, M\}$, define

$$\mathcal{J} = \{\, j \in \{1, \ldots, n\} : \; x_j(i) = 1 \,\}, \tag{23}$$

and denote

$$\mathcal{X}^*(\mathcal{J}) = \bigcup_{m=1}^{M} \{\mathbf{x}^*(m)\},$$

where $\mathbf{x}^*(m) = (x_1^*(m), \ldots, x_n^*(m))$ is the m-th codeword of the $\mathrm{ORN}_{(-,+)}$ code whose components are replaced with the $'*'$ symbol at positions that do not belong to $\mathcal{J}$, i.e., $x_j^*(m) = x_j(m)$ if $j \in \mathcal{J}$ and $x_j^*(m) = *$ if $j \notin \mathcal{J}$. Since the vectors $\mathbf{x}(1) \wedge \mathbf{x}(i), \ldots, \mathbf{x}(M) \wedge \mathbf{x}(i)$ are distinct, the n-tuples $\mathbf{x}^*(1), \ldots, \mathbf{x}^*(M)$ are also distinct and

$$\sum_{\mathbf{x}\in\mathcal{X}^*(\mathcal{J})} P(\mathbf{x}) = \sum_{m=1}^{M} P(\mathbf{x}^*(m)). \tag{24}$$

On the other hand,

$$\sum_{\mathbf{x}\in\mathcal{X}^*(\mathcal{J})} P(\mathbf{x}) \le \sum_{\mathbf{x}\in\mathcal{X}(\mathcal{J})} P(\mathbf{x}) \tag{25}$$

since $\mathcal{X}^*(\mathcal{J}) \subseteq \mathcal{X}(\mathcal{J})$. Thus, combining (22), (24), and (25) we obtain

$$\sum_{m=1}^{M} P(\mathbf{x}^*(m)) \le 1. \tag{26}$$

However

$$\begin{aligned}
\sum_{m=1}^{M} P(\mathbf{x}^*(m)) - 1 &= \frac{1}{M} \sum_{m=1}^{M} \left(\frac{P(\mathbf{x}^*(m))}{1/M} - 1 \right) \\
&\ge \frac{1}{M} \sum_{m=1}^{M} \ln \frac{P(\mathbf{x}^*(m))}{1/M} \\
&= \frac{1}{M} \sum_{m=1}^{M} \ln P(\mathbf{x}^*(m)) + \ln M,
\end{aligned}$$

where the inequality $z - 1 \geq \ln z$ was used. Using (19), (20), and (21) it is easy to see that

$$\begin{aligned}
\frac{1}{M}\sum_{m=1}^{M} \ln P(\mathbf{x}^*(m)) &= \frac{1}{M}\sum_{m=1}^{M} \ln \prod_{j\in\mathcal{J}} \gamma_j(x_j(m)) \\
&= \sum_{j\in\mathcal{J}} \frac{1}{M}\sum_{m=1}^{M} \ln \gamma_j(x_j(m)) \\
&= \sum_{j\in\mathcal{J}} \sum_{x\in\{0,1\}} \frac{1}{M} \left|\{m = 1,\ldots,M : x_j(m) = x\}\right| \ln \gamma_j(x) \\
&= \sum_{j\in\mathcal{J}} \sum_{x\in\{0,1\}} \gamma_j(x) \ln \gamma_j(x) \\
&= -\sum_{j\in\mathcal{J}} h(\gamma_j) \ln 2.
\end{aligned}$$

Hence

$$\sum_{m=1}^{M} P(\mathbf{x}^*(m)) - 1 \geq -\sum_{j\in\mathcal{J}} h(\gamma_j) \ln 2 + \ln M,$$

and using (26) we obtain

$$\log M \leq \sum_{j\in\mathcal{J}} h(\gamma_j) = \sum_{j: x_j(i)=1} h(\gamma_j), \tag{27}$$

where we returned the dependence on i (see (23)). Inequality (27) is valid for all $i \in \{1,\ldots,M\}$ and we may write

$$\log M \leq \frac{1}{M}\sum_{i=1}^{M} \sum_{j: x_j(i)=1} h(\gamma_j). \tag{28}$$

Since $|\{j : x_j(i) = 1\}| = w$ (all codewords have the Hamming weight w), we use the fact that $h(z)$ is a convex up function of z and continue (28) as follows :

$$\begin{aligned}
\log M &\leq w \cdot \frac{1}{Mw}\sum_{i=1}^{M} \sum_{j: x_j(i)=1} h(\gamma_j) \qquad (29) \\
&\leq w \cdot h\left(\frac{1}{Mw}\sum_{i=1}^{M} \sum_{j: x_j(i)=1} \gamma_j\right)
\end{aligned}$$

$$
\begin{aligned}
&= w \cdot h\left(\frac{1}{w}\sum_{j=1}^{n}\gamma_j \frac{1}{M}\sum_{i=1}^{M} x_j(i)\right) \\
&= w \cdot h\left(\frac{1}{w}\sum_{j=1}^{n}\gamma_j^2\right).
\end{aligned}
$$

The minimum of the sum $\gamma_1^2 + \ldots + \gamma_n^2$ taken over all $\gamma_1, \ldots, \gamma_n \in [0,1]$ such that $\gamma_1 + \ldots + \gamma_n = w$ (wM is the total number of 1's in the matrix) is attained when $\gamma_1 = \ldots = \gamma_n = w/n$ and

$$
\frac{1}{w}\sum_{j=1}^{n}\gamma_j^2 \geq \frac{n}{w}\cdot\frac{w^2}{n^2} = \frac{w}{n}.
$$

Since $h(z)$ is a decreasing function when $z \in [1/2, 1]$ and $w > n/2$, this inequality implies

$$
h\left(\frac{1}{w}\sum_{j=1}^{n}\gamma_j^2\right) \leq h\left(\frac{w}{n}\right).
$$

Thus, (18) follows from (29).

To prove Corollary 2 we note that any $\mathrm{ORN}_{(-,+)}$ code of length n for M messages has at most $n+1$ subcodes containing codewords of the Hamming weight $0, \ldots, n$. Therefore M is upper-bounded by a product of $n+1$ and the maximal number of messages that can be transmitted using an $\mathrm{ORN}_{(-,+)}$ code whose codewords have a fixed Hamming weight, and (18) leads to the inequality

$$
\begin{aligned}
\log M &\leq \log(n+1) + \max\left\{ \max_{0\leq w\leq n/2} w,\ \max_{n/2<w\leq n} w\cdot h(w/n) \right\} \\
&= \log(n+1) + \max_{n/2\leq w\leq n} w\cdot h(w/n) \\
&\leq \log(n+1) + n\cdot \max_{0\leq\gamma\leq 1} F_{(-)}(\gamma).
\end{aligned}
$$

□

3.3 Proof of direct statements

Considerations of Section 2 in the part devoted to the case (E_+, D_+) can be presented as a proof of the direct statement of the Theorem for that case.

Note also that, obviously, $R_{(+,+)} \geq R_{(+,-)}$ and we can restrict our attention to the case (E_+, D_-).

Proposition 3 [4] *If*

$$M \cdot |\mathcal{B}_w^n| \cdot \exp\left\{-\frac{1}{M}\sum_{i=0}^{n-w}\binom{w}{i}\right\} < 1, \tag{30}$$

then there exist $ORN_{(+,-)}$ codes of length n for M messages consisting of codewords of the Hamming weight at least w, where the set $\mathcal{B}_w^n$ is defined in (10).

Corollary 3 *There exist $ORN_{(+,-)}$ codes of length n and rate*

$$R = \max_{0 \leq \gamma \leq 1} F_{(+)}(\gamma) - \varepsilon_n,$$

where the function $F_{(+)}(\gamma)$ is defined in (17) and $\varepsilon_n \to 0$, as $n \to \infty$.

Proof For all binary vectors $\mathbf{y} \in \mathcal{B}_w^n$, let us denote the set of vectors $\mathbf{x} \in \mathcal{B}_w^n$ that cover $\overline{\mathbf{y}}$ (or, equivalently, such that $\mathbf{x} \vee \overline{\mathbf{y}} = \mathbf{x}$) by

$$\mathcal{C}_w^n(\overline{\mathbf{y}}) = \{\ \mathbf{x} \in \mathcal{B}_w^n :\ \mathbf{x} \geq \overline{\mathbf{y}}\ \}.$$

Then

$$|\mathcal{C}_w^n(\overline{\mathbf{y}})| = \sum_{k=w}^{n}\binom{n - w_H(\overline{\mathbf{y}})}{k - w_H(\overline{\mathbf{y}})} = \sum_{k=w}^{n}\binom{n - w_H(\overline{\mathbf{y}})}{n-k}$$

and since $\mathbf{y} \in \mathcal{B}_w^n$ implies $w_H(\overline{\mathbf{y}}) \leq n - w$,

$$|\mathcal{C}_w^n(\overline{\mathbf{y}})| \geq \sum_{k=w}^{n}\binom{w}{n-k} = \sum_{i=0}^{n-w}\binom{w}{i}. \tag{31}$$

Suppose that $\mathcal{X} = (\mathcal{X}(1), \ldots, \mathcal{X}(M))$ is a partition of the set $\mathcal{B}_w^n$, i.e., each vector $\mathbf{x} \in \mathcal{B}_w^n$ belongs to exactly one set $\mathcal{X}(m)$ and $\bigcup_{m=1}^{M}\mathcal{X}(m) = \mathcal{B}_w^n$. Let

$$\Lambda(\mathcal{X}) = \sum_{m=1}^{M}\sum_{\mathbf{y} \in \mathcal{B}_w^n}\lambda(\mathcal{X}(m), \mathbf{y}),$$

[4] Proposition 3 is of interest only when n is big enough. For example, if $n = 16$ then it guarantees existence of an $\mathrm{ORN}_{(+,-)}$ code with $R \approx 0.2715$ and if $n = 32$ then $R \approx 0.3435$ (the parameter w should be assigned as 11 and 23, respectively), while the $\mathrm{ORN}_{(+,-)}$ code of length $n = 5$ given in Table 4 has rate $R \approx 0.5170$.

where

$$\lambda(\mathcal{X}(m), \mathbf{y}) = \begin{cases} 1, & \text{if } \mathbf{x} \notin \mathcal{C}_w^n(\overline{\mathbf{y}}) \text{ for all } \mathbf{x} \in \mathcal{X}(m), \\ 0, & \text{otherwise.} \end{cases}$$

Obviously,

$$\Lambda(\mathcal{X}) = 0 \implies \lambda(\mathcal{X}(m), \mathbf{y}) = 0, \text{ for all } m = 1, \ldots, M \text{ and } \mathbf{y} \in \mathcal{B}_w^n. \tag{32}$$

For all $\mathbf{x} \in \mathcal{B}_w^n$, let us organize an independent random experiment with M outcomes that gives some element of the set $\{1, \ldots, M\}$ with probability $1/M$. If the outcome is m, then we extend the set $\mathcal{X}(m)$ adding the vector $\mathbf{x}$. As a result, the parameter $\Lambda(\mathcal{X})$ becomes a random variable having the expected value

$$\mathbb{E}\Lambda(\mathcal{X}) = \sum_{m=1}^{M} \sum_{\mathbf{y} \in \mathcal{B}_w^n} \mathbb{E}\lambda(\mathcal{X}(m), \mathbf{y}),$$

where

$$\mathbb{E}\lambda(\mathcal{X}(m), \mathbf{y}) = \left(1 - \frac{1}{M}\right)^{|\mathcal{C}_w^n(\overline{\mathbf{y}})|}. \tag{33}$$

The last equation follows from the observation that a particular $\mathbf{x} \in \mathcal{B}_w^n$ is not an element of $\mathcal{X}(m)$ with probability $1 - 1/M$ and there are $|\mathcal{C}_w^n(\overline{\mathbf{y}})|$ possibilities for the set $\mathcal{X}(m)$ to have at least one element covering a fixed $\overline{\mathbf{y}}$. However $\Lambda(\mathcal{X})$ is an integer for any partition $\mathcal{X}$ and

$$\mathbb{E}\Lambda(\mathcal{X}) < 1 \implies \exists \mathcal{X} : \Lambda(\mathcal{X}) = 0. \tag{34}$$

To complete the proof we estimate (33) using the inequality $1 - z \le e^{-z}$ and (31) :

$$\mathbb{E}\lambda(\mathcal{X}(m), \mathbf{y}) \le \exp\left\{-\frac{|\mathcal{C}_w^n(\overline{\mathbf{y}})|}{M}\right\} \le \exp\left\{-\frac{1}{M} \sum_{i=0}^{n-w} \binom{w}{i}\right\}.$$

Hence,

$$\mathbb{E}\Lambda(\mathcal{X}) \le M \cdot |\mathcal{B}_w^n| \cdot \exp\left\{-\frac{1}{M} \sum_{i=0}^{n-w} \binom{w}{i}\right\}.$$

This upper bound and statements (32), (34) imply (30).

To prove Corollary 3, let us assign $w = n\gamma$, where γ maximizes the expression (16). Then using (13) we obtain

$$\frac{1}{M} \sum_{i=0}^{n(1-\gamma)} \binom{n\gamma}{i} = 2^{n(\Delta R - \delta_n)},$$

where $\Delta R = R_{(+,-)} - R$, and using the trivial bounds $M, |\mathcal{B}_w^n| \le 2^n$ conclude that inequality (30) is valid when

$$\Delta R - \delta_n - \frac{\log \ln(2n)}{n} > 0.$$

Hence, we can assign ΔR as a function ε_n that tends to 0 as $n \to \infty$.

$\square$

4 Acknowledgment

The work was initiated by the lectures of Professor Rudolf Ahlswede on coding problems for different models of the memory. The author is also grateful to Professor Ahlswede and the other members of the group, Bernhard Balkenhol, Lars Bäumer, Ning Cai, Christian Deppe, Reinhard Franz, Levon Khachatrian, Christian Kleinewächter, and Ulrich Tamm for the interesting and helpful discussions.

Table 1: Coding for the ORN memory at time instant t.

Case	Encoding	Decoding
(E_+, D_+)	$\mathbf{x}_t = f(m_t, \mathbf{y}_{t-1})$	$m_t = \varphi(\mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1}, \mathbf{y}_{t-1})$
(E_+, D_-)	$\mathbf{x}_t = f(m_t, \mathbf{y}_{t-1})$	$m_t = \varphi(\mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1})$
(E_-, D_+)	$\mathbf{x}_t = f(m_t)$	$m_t = \varphi(\mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1}, \mathbf{y}_{t-1})$
$(E_-, D_-)'$	$\mathbf{x}_t = f(m_t)$, $m_t \neq m_{t-1}$	$m_t = \varphi(\mathbf{x}_t \vee \overline{\mathbf{y}}_{t-1})$

Table 2: Auxiliary code for the $\mathrm{ORN}_{(+,+)}$ code of length $n = 6$ for $M = 11$ messages; $R = (\log 11)/6 \approx 0.5766$.

$m =$	1	2	3	4	5	6
$\mathbf{c}(m) =$	0011	0101	0110	1001	1010	1100
$m =$	7	8	9	10	11	
$\mathbf{c}(m) =$	0111	1011	1101	1110	1111	

Table 3: Example of coding with the use of the $\mathrm{ORN}_{(+,+)}$ code of length $n = 6$ for $M = 11$ messages.

	Encoding			Storing		Decoding		
t	$\mathcal{O}'_{t-1}$	m_t	$\mathbf{x}_t$	$\mathbf{y}_t$	$\mathcal{O}_t$	$\mathcal{O}'_{t-1}$	$\mathbf{y}^*_t$	m_t
0				111111	$\emptyset$			
1	$\{1,2\}$	3	$11\mathbf{0110}$	110110	$\{3,6\}$	$\{1,2\}$	$**\mathbf{0110}$	3
2	$\{3,6\}$	6	$\mathbf{11}1\mathbf{00}1$	111001	$\{4,5\}$	$\{3,6\}$	$\mathbf{11}*\mathbf{00}*$	6
3	$\{4,5\}$	7	$\mathbf{011}11\mathbf{1}$	011111	$\{1\}$	$\{4,5\}$	$\mathbf{011}**\mathbf{1}$	7
4	$\{1,2\}$	1	$11\mathbf{0011}$	110011	$\{3,4\}$	$\{1,2\}$	$**\mathbf{0011}$	1
5	$\{3,4\}$					$\{3,4\}$		

Table 4: $\mathrm{ORN}_{(+,-)}$ code of length $n = 5$ for $M = 6$ messages; $R = (\log 6)/5 \approx 0.5170$.

$m =$	1	2	3	4	5	6
$\mathbf{X}(m) =$	11100 00111 11011	01110 10011 11101	11010 01101 10111	01011 10101 11110	11001 10110 01111	11111

Table 5: Example of coding with the use of the $\mathrm{ORN}_{(+,-)}$ code of length $n = 5$ for $M = 6$ messages.

	Encoding			Storing		Decoding
t	$\mathcal{O}_{t-1}$	m_t	$\mathbf{x}_t$	$\mathbf{y}_t$	$\mathcal{O}_t$	m_t
0				11111	$\emptyset$	
1	$\emptyset$	3	11010	11010	$\{3,5\}$	3
2	$\{3,5\}$	3	01**1**0**1**	01101	$\{1,4\}$	3
3	$\{1,4\}$	1	**1**10**1**1	11011	$\{3\}$	1
4	$\{3\}$	2	01**1**10	01110	$\{1,5\}$	2
5	$\{1,5\}$					

Table 6: $\mathrm{ORN}_{(+,-)}$ code of length $n = 11$ for $M = 58$ messages; $R = (\log 58)/11 \approx 0.5325$ (the codewords that form the matrices $\mathbf{X}(m)$ are given in the octal representation).

m	$\mathbf{X}(m)$				m	$\mathbf{X}(m)$			
1	3273	3547	3734	777	30	1637	2375	3762	3557
2	2557	3633	3764	1377	31	1373	2756	3635	3567
3	2772	3365	3617	1577	32	1717	2765	3276	3573
4	2177	3707	3770	1677	33	1376	2753	3627	3575
5	2673	3355	3566	1737	34	1727	2373	3655	3576
6	2277	3565	3732	1757	35	1753	2774	3167	3637
7	2676	3317	3571	1767	36	1177	2771	3726	3657
8	2576	3257	3725	1773	37	1375	2573	3716	3667
9	2747	3173	3636	1775	38	1575	2736	3347	3673
10	2667	3333	3555	1776	39	1567	2733	3356	3675
11	1735	3473	3746	2377	40	1337	2567	3751	3676
12	1756	3275	3723	2577	41	776	3371	3467	3717
13	1367	3533	3754	2677	42	1771	2657	3176	3727
14	1774	3077	3743	2737	43	1747	2477	3374	3733
15	1772	3267	3535	2757	44	1667	2376	3553	3735
16	1755	3137	3672	2767	45	577	3353	3665	3736
17	1657	3327	3574	2773	46	1675	2337	3572	3747
18	1277	3527	3752	2775	47	1736	2367	3475	3753
19	1763	3335	3457	2776	48	1576	2763	3237	3755
20	767	3656	3731	3177	49	377	3437	3761	3756
21	1766	2575	3713	3277	50	1557	2675	3336	3763
22	773	3476	3745	3337	51	677	3372	3517	3765
23	775	3536	3663	3357	52	1537	2357	3671	3766
24	1673	2735	3556	3367	53	1676	2727	3157	3771
25	1477	2766	3715	3373	54	737	3175	3647	3772
26	1573	2717	3666	3375	55	3563	2637	1357	3774
27	1765	2537	3653	3376	56	2777	1777	3377	3577
28	1733	2755	3366	3477	57	3677	3737	3757	3767
29	757	3363	3674	3537	58	3777			

Table 7: The known $\mathrm{WUM}_{(+,-)}$ / $\mathrm{ORN}_{(+,-)}$ codes.

n	M	R	Reference
5	6	0.5170	Willems and Vinck [5]
10	41	0.5376	Ahlswede and Zhang [13]
11	58	0.5325	Simonyi [10] + Table 6
14	213	0.5525	Koschnick [14]
15	307	0.5508	Ahlswede and Zhang [13]
17	767	0.5637	Koschnick [14]

Table 8: Example of coding with the use of the $\mathrm{ORN}_{(-,+)}$ code consisting of $M = 3$ codewords (011, 101, 110) of length $n = 3$; $R = (\log 3)/3 \approx 0.5283$.

	Encoding		Storing		Decoding		
t	m_t	$\mathbf{x}_t$	$\mathbf{y}_t$	$\mathcal{O}_t$	$\mathcal{O}_{t-1}$	$\mathbf{y}_t^*$	m_t
0			111	$\emptyset$			
1	3	110	110	$\{3\}$	$\emptyset$	**110**	3
2	3	110	111	$\emptyset$	$\{3\}$	**11**$*$	3
3	1	011	011	$\{1\}$	$\emptyset$	**011**	1
4	2	101	101	$\{2\}$	$\{1\}$	$*$**01**	2
5					$\{2\}$		

Table 9: $\mathrm{ORN}_{(-,+)}$ code of length $n = 12$ for $M = 36$ messages constructed using the Shearer's algorithm; $R = (\log 36)/12 \approx 0.4308$.

m	$\mathbf{m}$	$\mathbf{x}(m)$	m	$\mathbf{m}$	$\mathbf{x}(m)$
1	$\sqcup, 0, 0, 0$	000 011 011 011	19	$0, 0, \sqcup, 0$	011 011 000 011
2	$\sqcup, 0, 1, 2$	000 011 101 110	20	$0, 1, \sqcup, 2$	011 101 000 110
3	$\sqcup, 0, 2, 1$	000 011 110 101	21	$0, 2, \sqcup, 1$	011 110 000 101
4	$\sqcup, 1, 0, 2$	000 101 011 110	22	$1, 0, \sqcup, 2$	101 011 000 110
5	$\sqcup, 1, 1, 1$	000 101 101 101	23	$1, 1, \sqcup, 1$	101 101 000 101
6	$\sqcup, 1, 2, 0$	000 101 110 011	24	$1, 2, \sqcup, 0$	101 110 000 011
7	$\sqcup, 2, 0, 1$	000 110 011 101	25	$2, 0, \sqcup, 1$	110 011 000 101
8	$\sqcup, 2, 1, 0$	000 110 101 011	26	$2, 1, \sqcup, 0$	011 101 000 011
9	$\sqcup, 2, 2, 2$	000 110 110 110	27	$2, 2, \sqcup, 2$	110 110 000 110
10	$0, \sqcup, 0, 0$	011 000 011 011	28	$0, 0, 0, \sqcup$	011 011 011 000
11	$0, \sqcup, 1, 2$	011 000 101 110	29	$0, 1, 2, \sqcup$	011 101 110 000
12	$0, \sqcup, 2, 1$	011 000 110 101	30	$0, 2, 1, \sqcup$	011 110 101 000
13	$1, \sqcup, 0, 2$	101 000 011 110	31	$1, 0, 2, \sqcup$	101 011 110 000
14	$1, \sqcup, 1, 1$	101 000 101 101	32	$1, 1, 1, \sqcup$	101 101 101 000
15	$1, \sqcup, 2, 0$	101 000 110 011	33	$1, 2, 0, \sqcup$	101 110 011 000
16	$2, \sqcup, 0, 1$	110 000 011 101	34	$2, 0, 1, \sqcup$	110 011 101 000
17	$2, \sqcup, 1, 0$	110 000 101 011	35	$2, 1, 0, \sqcup$	011 101 011 000
18	$2, \sqcup, 2, 2$	110 000 110 110	36	$2, 2, 2, \sqcup$	110 110 110 000

Table 10: Example of coding with the use of the $\mathrm{ORN}'_{(-,-)}$ code consisting of $M = 3$ codewords (011, 101, 110) of length $n = 3$; $R = (\log 2)/3 \approx 0.3333$.

	Encoding		Storing		Decoding
t	m_t	$\mathbf{x}_t$	$\mathbf{y}_t$	$\mathcal{O}_t$	m_t
0			111	$\emptyset$	
1	3	110	110	$\{3\}$	3
2	2	101	101	$\{2\}$	2
3	1	011	011	$\{1\}$	1
4	2	101	101	$\{2\}$	2

$$
\begin{tikzcd}
m_1, m_2, \ldots \in \{1, \ldots, M\} \arrow[r, "\text{Encoding}"] & \mathbf{x}_1, \mathbf{x}_2, \ldots \in \{0,1\}^n \arrow[d, "\text{Storing}"] \\
m_1, m_2, \ldots & \mathbf{y}_0 = 1^n, \mathbf{y}_1 = \mathbf{x}_1, \mathbf{y}_2 = \mathbf{x}_2 \vee \overline{\mathbf{y}}_1, \ldots \arrow[l, "\text{Decoding}"']
\end{tikzcd}
$$

Figure 1: Uniquely decodable coding scheme for the ORN memory.

References

[1] A.V.Kuznetsov and B.S.Tsybakov, "Coding in a memory with defective cells," *Problemy Peredachi Informatsii,* vol.10, no.2, pp.52-60, Apr.-June 1974.
English translation : *Probl.Inform.Transmission,* pp.132-138.

[2] R.L.Rivest and A.Shamir, "How to reuse a write–once memory," *Inform. and Control,* vol.55, pp.1-19, 1982.

[3] J.K.Wolf, A.D.Wyner, J.Ziv, and J.Körner, "Coding for write–once memory," *AT and T Bell Lab. Tech. J.,* vol.63, no.6, pp.1084-1112, July - Aug. 1984.

[4] J.M.Borden, "Coding for write–unidirectional memories," Preprint, 1984.

[5] F.M.J.Willems and A.J.Vinck, "Repeated recordings for an optical disk," in *Proceedings, 7th Symposium on Information Theory in the Benelux,* Delft University Press, pp.49-53, 1986.

[6] R.Ahlswede and Z.Zhang, "Coding for write–efficient memory," *Inform. and Comput.,* vol.83, no.1, pp.80-97, 1989.

[7] G.Cohen and G.Zemor, "Write–isolated memories (WIMs)," *Discrete Math.,* vol.114, pp.105-113, 1993.

[8] G.Cohen, "A survey : Writing on some binary memories with constraints (W*M's)," in *CISM Courses and Lectures 313, "Geometries, codes and cryptography",* Springer-Verlag, pp.213-227, 1990.

[9] W.Stallings, *Computer Organization and Architecture,* New York: Macmillan, 1987.

[10] G.Simonyi, "On write–unidirectional memory codes," *IEEE Trans.Inform. Theory,* vol.35, no.3, pp.663-667, May 1989.

[11] R.Ahlswede and I.Csiszár, "To get a bit of information may be as hard as to get full information," *IEEE Trans. Inform. Theory,* vol.27, no.4, pp.398-408, July 1981.

[12] R.M.Fano, *Transmission of Information,* Cambridge : MIT Press, 1961.

[13] R.Ahlswede and Z.Zhang, "On multi–user write–efficient memories," *IEEE Trans. Inform. Theory,* vol.40, no.3, pp.674-686, May 1994.

[14] K.-U.Koschnick, "Good code design with combinatorial approximation algorithms," *SFB 343 "Diskrete Strukturen in der Mathematik,"* Preprint 91-017, Universität Bielefeld, 1991.

[15] P.Frankl and Z.Füredi, "Union-free hypergraphs and probability theory," *European Journal of Combinatorics,* vol.5, pp.127-131, 1984.

[16] J.B.Shearer, "A new construction for cancellative families of sets," *The Electronic Journal of Combinatorics,* vol.3, 1996.

[17] L.Lovász, *Combinatorial Problems and Exercises,* Amsterdam : North-Holland, 1979.