WOM Codes with Uninformed Encoder

Michal Horovitz and Eitan Yaakobi

Computer Science Department
Technion, Haifa, Israel

26/04/2015
Coding Theory Seminar
Outline

- Write-Once-Memory (WOM) motivation
- WOM models - definitions
- Previous results:
  - capacity region and maximum sum-rate
- The EU:DU model
  - Binary 2-write - simple construction
  - Binary t-write - recursive construction
  - Capacity achieving Binary 2-write, reduction to the Z channel
  - Non-Binary t-write
- The EU:DI model
  - Binary 2-write - simple construction
  - Capacity achieving Binary 2-write, reduction to the erasure channel
  - Non-Binary t-write
A nonvolatile memory both electrically programmable and electrically erasable.

Adding charge to a single cell is a fast operation.

Removing charge from a single cell requires erasing an entire block (10^6 cells).

Multilevel Flash Memory

Each cell stores one of \( q \) levels and can be regarded as a symbol over a discrete alphabet of size \( q \).
Write-Once-Memory (WOM) Codes

Problem:
Cannot rewrite the memory without an erasure

Solution:
Write-Once-Memory (WOM) Codes.
One of the most efficient schemes to decrease the number of block erasures

Introduced by Rivest and Shamir,
“How to reuse a write-once memory”, 1982
Write-Once-Memories (WOM)
Rivest and Shamir, 1982

Ex: 2-write binary WOM code

<table>
<thead>
<tr>
<th>Bits Value</th>
<th>1st Write</th>
<th>2nd Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>000</td>
<td>111</td>
</tr>
<tr>
<td>01</td>
<td>001</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>010</td>
<td>101</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>011</td>
</tr>
</tbody>
</table>

\[ R_1 = R_2 = \frac{2}{3} \]

\[ R^{sum} = 1.33 \]
The goal:
Maximize the total number of bits which are possible to be written to the memory in $t$ writes,
while preserving the property that cells can only increase their level.

There are in fact four models of WOM, depending whether the encoder and decoder are informed with the memory state before encoding on each write.

J.K.Wolf, A.D.Wyner, J. Ziv, and J.Korner,
“Coding for a write-once memory,”
EI - Encoder Informed
The encoder read the memory before encoding new data.

EU - Encoder Uninformed
The encoder doesn't read the memory before encoding new data.

DI - Decoder Informed
The decoder knows the memory state before the new data was encoded.

DU - Decoder Uninformed
The decoder doesn't know the memory state before the new data was encoded. The decoder knows only the last state.

Example:
First write: 0011
Second write: 0101
Memory state after 1st write: \( c_1 = 0011 \)
Memory state after 2nd write: \( c_2 = 0111 \)

\( m \) - the encoded message on the 2nd write.

The input of the encoder on the 2nd write:

- EI: \( c_1, m \)
- EU: \( m \)

The input of the decoder on the 2nd write:

- DI: \( c_1, c_2 \)
- DU: \( c_2 \)
WOM - Models

EI:DI
The easiest to solve.
Encoder and decoder know the previous state.

EI:DU
The conventional model.
Only the encoder knows the previous state.
Rivest and Shamir ‘82, Merkx ‘84, Cohen, Godlewski, and Merkx ‘86,
Wu and Jiang ‘09, Wu ‘10,
Burshtein and Strugatski ’12, Yaakobi and Shpilka ’12

EU:DI
Only the decoder knows the previous state.

EU:DU
The hardest to solve.
Encoder and decoder don't know the previous state.
An \([n, t; M_1, \ldots, M_t]_q^{EX:DY}\) WOM code

- **n** \(n\) number of cells
- **t** \(t\) number of writes
- **\(M_i\)** number of messages written in the \(i\)th write.
- **q** number of charge-levels
- **EX: DY** the WOM model, \(X, Y \in \{I, U\}\).

The rate on \(i\)-th write: \(R_i = \frac{\log_2 M_i}{n}\)

The sum-rate: \(R_{\text{sum}} = \sum_{i=1}^{t} R_i\)

The capacity region, \(C_t\): the set of all achievable rate tuples of a \(t\)-write WOM.
The EI:DI, EI:DU, EU:DI models:

The capacity region, $C_t$

$$C_t = \left\{ (R_1, \ldots, R_t) | R_1 \leq h(p_1), R_2 \leq h(p_2)(1-p_1), \ldots, R_{t-1} \leq h(p_{t-1}) \prod_{i=1}^{t-2}(1-p_i), R_t \leq \prod_{i=1}^{t-1}(1-p_i), \right\}$$

the maximum sum-rate: $\log_2(t + 1)$

The EU:DU model:

The following region, $\hat{C}_t$ was shown to be achievable:

$$\hat{C}_t = \left\{ (R_1, \ldots, R_t) | R_1 \leq h(p_1), R_2 \leq h(p_1 p_2) - p_2 h(p_1), R_3 \leq h(p_1 p_2 p_3) - p_3 h(p_1 p_2), \ldots, R_t \leq h(\prod_{j=1}^{t} p_j) - p_t h(\prod_{j=1}^{t-1} p_j), \right\}$$

where $0 \leq p_1, \ldots, p_t \leq 1$.

the maximum sum-rate:

$$P_t = \sup_{0 \leq p_1, \ldots, p_t \leq 1} \left\{ h(\prod_{j=1}^{t} p_j) + \sum_{i=2}^{t} \left( (1 - p_i) h(\prod_{j=1}^{i-1} p_j) \right) \right\} \leq \frac{\pi^2}{6 \ln 2} \approx 2.37$$
**Binary 2-Write WOM Capacity**

The **EI:DI, EI:DU, EU:DI** models

The capacity region, $C_t$

The **EU:DU** model

The achievable region, $\hat{C}_t$

The maximum sum-rate:

$$P_t = \sup_{0 \leq p_1, \ldots, p_t \leq 1} \left\{ h \left( \prod_{j=1}^{t} p_j \right) + \sum_{i=2}^{t} \left( 1 - p_i \right) h \left( \prod_{j=1}^{i-1} p_j \right) \right\}$$

$$\log_2 (t + 1)$$
Max Sum-Rate of Binary $t$-Write WOM

<table>
<thead>
<tr>
<th>$t$</th>
<th>Three models $\log_2(t + 1)$</th>
<th>EU:DU model $P_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.5849</td>
<td>1.3881</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1.600</td>
</tr>
<tr>
<td>4</td>
<td>2.3219</td>
<td>1.7356</td>
</tr>
<tr>
<td>5</td>
<td>2.5849</td>
<td>1.9695</td>
</tr>
<tr>
<td>$t \to \infty$</td>
<td>$\log_2(t + 1)$</td>
<td>$P_t \leq \frac{\pi^2}{6\ln2} \approx 2.37$</td>
</tr>
</tbody>
</table>
Outline

- Write-Once-Memory (WOM) motivation
- WOM models - definitions
- Previous results:
  - capacity region and maximum sum-rate
- The EU:DU model
  - Binary 2-write - simple construction
  - Binary t-write - recursive construction
  - Capacity achieving Binary 2-write, reduction to the Z channel
  - Non-Binary t-write
- The EU:DI model
  - Binary 2-write - simple construction
  - Capacity achieving Binary 2-write, reduction to the erasure channel
  - Non-Binary t-write
Binary 2-Write WOM EU:DU
a Simple Construction

Construction of a $[2,2; 3,2]^{EU:DU}_2$ WOM code.

- 2 number of cells.
- 2 number of writes.
- 3,2 number of messages written on the first and second writes, respectively.
- 2 number of charge levels.

**First write:**
a ternary symbol is written:
0 $\mapsto (0,0)$, 1 $\mapsto (0,1)$, 2 $\mapsto (1,0)$

**Second write:**
one more bit is written.
0 $\mapsto (0,0)$ - no change, 1 $\mapsto (1,1)$

$$R_{2}^{\text{sum}} = \frac{\log_2 3 + 1}{2} \approx 1.29$$

<table>
<thead>
<tr>
<th>1st Write</th>
<th>2nd Write</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>
**Binary 3-Write WOM EU:DU**

**Given:** \([2,2; 3,2]_{2}^{E:DU}\) WOM code, \(C_2\) \(R_{2}^{sum} = \frac{\log_2 3+1}{2} \approx 1.29\)

**Construct:** \([4,3; 3^2, 3, 2]_{2}^{E:DU}\) WOM code, \(C_3\) \(R_{3}^{sum} = \frac{\log_2 3}{2} + \frac{\log_2 3+1}{4} \approx 1.43\)

The 4 cells are partitioned into 2 pairs: 

First write: A ternary symbol is written in each pair:

\(0 \mapsto (0,0), 1 \mapsto (0,1), 2 \mapsto (1,0)\)

Second–third writes: Each **pair of cells** represents one **logical cell**:

\((1, 1) \leftrightarrow 1, (0, 0), (0, 1), (1, 0) \leftrightarrow 0\).

Apply \(C_2\) for the 2 logical cells:

Write: \(0 \rightarrow\) the pair is unchanged, \(1 \rightarrow\) program \((1, 1)\).

Read \((0, 0), (0, 1), (1, 0) \rightarrow 0, (1, 1) \rightarrow 1\).

**Ex1:** 1\(^{st}\) write: \(20 \in \{0,1,2\}^2\) →

1 | 0 | 0 | 0
→

0 | 0

2\(^{nd}\) write \(2 \in \{0,1,2\} →\)

1 | 0
→

1 | 1 | 0 | 0

3\(^{rd}\) write: \(1 \in \{0,1\}\) →

1 | 1
→

1 | 1 | 1 | 1
Given: \([2,2; 3,2]^{EU:DU}_2\) WOM code, \(C_2\) \(R^\text{sum}_2 = \frac{\log_2 3 + 1}{2} \approx 1.29\)

Construct: \([4,3; 3^2,3,2]^{EU:DU}_2\) WOM code, \(C_3\) \(R^\text{sum}_3 = \frac{\log_2 3}{2} + \frac{\log_2 3 + 1}{4} \approx 1.43\)

The 4 cells are partitioned into 2 pairs:

First write: A ternary symbol is written in each pair:

\[0 \leftrightarrow (0,0), 1 \leftrightarrow (0,1), 2 \leftrightarrow (1,0)\]

Second–third writes: Each pair of cells represents one logical cell:

\((1, 1) \leftrightarrow 1, (0, 0), (0, 1), (1, 0) \leftrightarrow 0.\)

Apply \(C_2\) for the 2 logical cells:

Write: \(0 \rightarrow\) the pair is unchanged, \(1 \rightarrow\) program \((1, 1)\).

Read \((0, 0), (0, 1), (1, 0) \rightarrow 0, (1, 1) \rightarrow 1.\)

Ex 1: 1\(^{st}\) write: \(12 \in \{0,1,2\}^2 \rightarrow \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array}\) → \(\begin{array}{cccc} 0 & 0 \end{array}\)

2\(^{nd}\) write \(1 \in \{0,1,2\} \rightarrow \begin{array}{cc} 1 & 0 \end{array}\) → \(\begin{array}{cccc} 1 & 1 & 0 & 0 \end{array}\)

3\(^{rd}\) write: \(0 \in \{0,1\}\) → no–change

\(\begin{array}{cccc} 1 & 1 & 1 & 1 \end{array}\)
Given: \([4,3; 3^2, 3, 2]_{EU:DU}^WOM\) code, \(C_3\) \(R_{3}^{sum} = \frac{\log_2 3}{2} + \frac{\log_2 3 + 1}{4} \approx 1.43\)

Construct: \([8,4; 3^4, 3^2, 3, 2]_{EU:DU}^WOM\) code, \(C_4\)

\(R_{4}^{sum} = \frac{\log_2 3}{2} + \frac{\log_2 3}{3} + \frac{\log_2 3 + 1}{8} \approx 1.64\)

The 8 cells are partitioned into 4 pairs.

First write: A ternary symbol is written in each pair:

\(0 \mapsto (0,0), 1 \mapsto (0,1), 2 \mapsto (1,0)\)

Second–fourth writes:

Each pair of cells represents one logical cell:

Apply \(C_3\) for the 4 logical cells.

Ex: 1\(^{st}\) write: \(0212 \in \{0,1,2\}^4\)

2\(^{nd}\) write: \(21 \in \{0,1,2\}^2\)

3\(^{rd}\) write: \(2 \in \{0,1,2\}\)

4\(^{th}\) write: \(1 \in \{0,1\}\)
Given: $C_t$ – an $[n, t; M_1, M_2, ..., M_t]^{EU:DU}_2$ WOM code.

Construct: $C_{t+1}$ – a $[2n, t + 1; 3^n, M_1, M_2, ..., M_t]^{EU:DU}_2$ WOM code.

The $2n$ cells are partitioned into pairs:

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First write: A ternary symbol is written in each pair:

$0 \leftrightarrow (0,0), 1 \leftrightarrow (0,1), 2 \leftrightarrow (1,0)$

Second–last writes:

Each **pair of cells** represents one logical cell:

$(1, 1) \leftrightarrow 1, (0, 0), (0, 1), (1, 0) \leftrightarrow 0$.

Apply $C_t$ for the $n$ logical cells.
Conclusion:

Given:
Binary 2-write WOM in the EU:DU model, with sum-rate: $R_{2}^{sum}$.

Construct:
Binary $t$-write WOM in the EU:DU model, with sum-rate:

$$R_{t}^{sum} = \log_{2} 3 \cdot \sum_{i=1}^{t-2} 2^{-i} + 2^{2-t} \cdot R_{2}^{sum}$$

Proof:
By applying the former construction recursively, and using a two-write WOM code with sum-rate $R_{2}^{sum}$ as the initial parameter of the recursion.

$$R_{t}^{sum} \rightarrow \log_{2} 3 \approx 1.58 \text{, as } t \rightarrow \infty.$$  

Recall: max sum rate is: 2.37
Motivation:

The next construction is based on a reduction to the Z channel. It achieves the rates-region $\hat{C}_t$ as long as we have capacity achieving codes for the Z channel.
The Z Channel Definitions

- The Z channel:
  - binary inputs and outputs
  - asymmetric errors: a zero can change to a one (with probability \( p \)), and not vice versa.

- The capacity of the Z channel:
  \[ h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p), \]
  where \( \alpha \) is the probability of occurrence 1 in the codewords.

- An \((n, M, \tau)_Z\) asymmetric-error-correcting code:
  - \( n \) - number of cells
  - \( M \) - number of codewords
  - \( \tau \) - number of asymmetric-errors which can be corrected
Reduction to the Z Channel

**Problem:** find capacity achieving codes in the Z channel...

**Examples:**

<table>
<thead>
<tr>
<th>(n, M, τ) z</th>
<th>M₁</th>
<th>M₂</th>
<th>Sum-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5,2,4) z</td>
<td>31</td>
<td>2</td>
<td>1.1908</td>
</tr>
<tr>
<td>(6,2,4) z</td>
<td>57</td>
<td>2</td>
<td>1.1388</td>
</tr>
<tr>
<td>(6,4,3) z</td>
<td>42</td>
<td>4</td>
<td>1.2320</td>
</tr>
<tr>
<td>(2,2,1) z</td>
<td>3</td>
<td>2</td>
<td>1.29</td>
</tr>
</tbody>
</table>

The simple construction $C = \{(00), (11)\}$
**Binary 2-Write WOM EU:DU Reduction to the Z Channel**

**Given:** $C$ - an $(n, M, \tau)_Z$ asymmetric-error-correcting code.

**Construct:** An $[n, 2; M_1, M]_{EU:DU}$ WOM code, where $M_1 = \sum_{i=0}^{\tau} \binom{n}{i}$.

1\textsuperscript{st} write: $M_1$ messages are written by programming at most $\tau$ cells.

2\textsuperscript{nd} write: the codewords of $C$ are written.
Given: capacity achieving codes in the Z channel.

Construct: an \([n, 2; 2^{nR'_1}, 2^{nR'_2}]_{EU:DU}^{EU:DU}\) WOM

for any \((R_1, R_2) \in \hat{C}_2, \; \epsilon > 0, \; R'_1 \geq R_1 - \epsilon, \; R'_2 \geq R_2 - \epsilon\)

Let \(p_1, p_2 \in [0, 1] : R_1 \leq h(p_1)\) and \(R_2 \leq h(p_1p_2) - p_2 h(p_1)\).

Let \(p = 1 - p_1\) (the error probability),
\(\alpha = 1 - p_2\) (probability of occurrence of 1),

and \(C\) be \((n, M, \tau)_{Z}\) asymmetric-error-correcting code, \(\tau \approx pn\) and
\[
\frac{\log_2 M}{n} \geq h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p) - \epsilon
\]

Previous construction: An \([n, 2; \Sigma_i^\tau \binom{n}{i}, M]_{EU:DU}^{EU:DU}\) WOM code.

\(R'_1 = \frac{\log_2 \Sigma_i^\tau \binom{n}{i}}{n} \geq h\left(\frac{\tau}{n}\right) - \epsilon \geq h(p) - \epsilon \geq R_1 - \epsilon\)

\(R'_2 = \frac{\log_2 M}{n} \geq h((1 - \alpha)(1 - p)) - (1 - \alpha)h(p) - \epsilon \geq R_2 - \epsilon\)
Generalize the reduction to Z channel:

- Non-binary asymmetric errors:
  \[ b \to \{ b + 1, b + 2, \ldots, q - 1 \} \]

- An \((n, M, \tau)_q\) asymmetric-error-correcting code,
  \(n\) - number of cells
  \(M\) - number of codewords
  \(\tau\) - any asymmetric error vector with Manhattan weight at most \(\tau\), can be corrected.

(the Manhattan weight of \(u \in [q]^n\) is \(w_M(u) = \sum_{i=0}^{n-1} u_i\))
Non-Binary 2-Write WOM EU:DU

Given: \( C \) - an \((n, M, \tau)_q\) asymmetric-error-correcting code.

Construct: An \([n, 2; M_1, M]_2^{EU:DU}\) WOM code, where \( M_1 \) is the number of \( q \)-ary length-\( n \) vectors of Manhattan weight at most \( \tau \).

1st write: programming vectors with Manhattan weight at most \( \tau \)

2nd write: the codewords of \( C \) are written.
Non-Binary WOM EU models

Upper Bounds

Maximum sum-rate for a $q$-ary $t$-write WOM for EI:DU model:

$$\log_2 \binom{t + q - 1}{t}$$

This is also an upper bound on the sum-rate of all models.

New upper bound for EU models:

$$(q - 1)R,$$

where $R$ is an upper bound of a binary $t$-write WOM in the same model.

The second is tighter for large $t$ (independent on $t$).
Given: $C_q$, an $[n, t; M_1, \ldots, M_t]^{EU:DX}_q$ WOM code

$$ \text{sum-rate: } R^{sum} = \frac{\sum_{i=1}^{t} \log_2 M_i}{n} $$

Construct: $C_2$, an $[(q-1)n, t; M_1, \ldots, M_t]^{EU:DX}_2$ WOM code

$$ \text{sum-rate: } R'^{sum} = \frac{\sum_{i=1}^{t} \log_2 M_i}{n(q-1)} = \frac{R^{sum}}{q-1} $$

Proof: An invertible function:

$$ f: [q] \rightarrow \{0^{q-1-k}1^k : k \in [q]\} $$

$$ k \mapsto 0^{q-1-k}1^k $$

The $i^{th}$ wire:

Encoder: $m \mapsto f(E_i(m))$, Decoder: $c \mapsto D_i(f^{-1}(c))$

$E_i/D_i$ are the encoding/decoding functions of the $i^{th}$ write of $C_q$, respectively.
Outline

- Write-Once-Memory (WOM) motivation
- WOM models - definitions
- Previous results:
  - capacity region and maximum sum-rate
- The EU:DU model
  - Binary 2-write - simple construction
  - Binary t-write - recursive construction
  - Capacity achieving Binary 2-write, reduction to the Z channel
  - Non-Binary t-write
- The EU:DI model
  - Binary 2-write - simple construction
  - Capacity achieving Binary 2-write, reduction to the erasure channel
  - Non-Binary t-write

- 3 number of cells.
- 2 number of writes.
- 4,4 number of messages written on the first, second writes, respectively.
- 2 number of charge levels.

<table>
<thead>
<tr>
<th>Message</th>
<th>First write</th>
<th>Second write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>011</td>
</tr>
</tbody>
</table>

$R_2^{sum} = 1.33$
Motivation:

The next construction is based on a reduction to the erasure channel. It achieves the capacity-region $C_t$ as long as we have capacity achieving codes for the erasure channel.
The binary erasure channel (BEC):
- binary inputs and outputs
- Errors-erasures: a bit value can be erased.

The capacity of the Z channel: $1 - p$

An $(n, M, r)$ erasure-correcting code:
- $n$ - number of cells
- $M$ - number of codewords
- $r$ - number of erasures which can be corrected

E.g. codes with minimum Hamming distance $r + 1$. 

Binary 2-Write WOM EU:DI Reduction to the Erasure Channel

**Given:** $C$ - an $(n, M, r)$ erasure-correcting code.

**Construct:** An $[n, 2; M_1, M]^{EU:DI}_2$ WOM code, where $M_1 = \sum_{i=0}^{r} \binom{n}{i}$.

1st write: $M_1$ messages are written by programming at most $r$ cells.

2nd write: the codewords of $C$ are written.

Example: the previous simple construction:
a $(3,4,1)$ erasure-correcting code:

$$C = \{(000), (110), (101), (011)\}.$$ 

**Given:** capacity achieving codes in the BEC.

**Construct:** an $[n, 2; 2^{nR'_1}, 2^{nR'_2}]_{EU:DI}^2$ WOM

for any $(R_1, R_2) \in C_2$, $\epsilon > 0$, $R'_1 \geq R_1 - \epsilon$, $R'_2 \geq R_2 - \epsilon$
Generalize the reduction to erasure channel:

Non-binary asymmetric erasure:

\[ b \rightarrow \{b, b + 1, b + 2, \ldots q - 1\} \] with '?'.

Thus, if \( a \in [q] \) is received with '?', then the correct value is in \([a]\).

**Given:** \( C \), a length-\( n \) code over \( q \)-ary symbols with Manhattan distance \( d \) and \( M \) codewords.

**Construct:** an \([n, 2; M_1, M]^{EU:DI}_q\), where \( M_1 \) is the number of \( q \)-ary length-\( n \) vectors of Manhattan weight at most \( d - 1 \).
We have seen WOM codes for the EU models:

The EU: **DU** model:

- Binary 2-write - simple construction $R = 1.29$ (max: 1.388)
- Binary t-write - recursive construction $R \rightarrow \log_2 3 \approx 1.58$
- Capacity achieving Binary 2-write,
  reduction to the Z channel.
  Generalization: Non-Binary t-write.

- Upper bounds

The EU: **DI** model:

- Binary 2-write - simple construction $R = 1.33$ (max: 1.584)
- Capacity achieving Binary 2-write,
  reduction to the binary erasure channel.
  Generalization: Non-Binary t-write.

Thank You