How to Reuse a "Write - Once" Memory†

(Preliminary Version)

Ronald L. Rivest
MIT Laboratory for Computer Science, Cambridge, Mass.
Adi Shamir
Weizmann Institute of Science, Rehovot, Israel

"Trouble him not; his wits are gone."
King Lear, III.vi.89

Abstract

Storage media such as digital optical disks, PROMS, or paper tape consist of a number of "write-once" bit positions (*wits*); each wit initially contains a "0" that may later be *irreversibly* overwritten with a "1". We demonstrate that such "write-once memories" (*woms*) can be "rewritten" to a surprising degree. For example, only 3 wits suffice to represent any 2-bit value in a way that can later be updated to represent any other 2-bit value. For large k, $1.29\ldots \cdot k$ wits suffice to represent a k-bit value in a way that can be similarly updated. Most surprising, allowing t writes of a k-bit value requires only $t + o(t)$ wits, for any fixed k. For fixed t, approximately $k \cdot t/\log(t)$ wits are required as $k \to \infty$. An n-wit WOM is shown to have a "capacity" (i.e. $k \cdot t$ when writing a k-bit value t times) of up to $n \cdot \log(n)$ bits.

† This research was supported in part by NSF grant MCS-8006938.

I. Introduction

Digital optical disks (a variation of the "video disks" used to store analog video data) are an exciting new storage medium. A single 12-inch disk costing $100 can be used to store over 10^{11} bits of data – the equivalent of 40 reels of magnetic tape – and to provide access to any of it in 1/10 second. Such an order-of-magnitude improvement in the cost/performance of memory technology can have dramatic effects. (See [Bu80], [Mc81], [Go82].)

However, such capability is achieved at the cost of making the writing process irreversible. The disks are used as follows. Each disk is manufactured with a thin reflective coating of tellurium. To write on the disk, a laser is used to melt submicron pits in the tellurium at specified positions, changing those positions from their virgin "0" state to a "1" state. To read the disk, the laser (at low power) illuminates each position on the disk; the lower reflectivity of the pits is easily sensed.

The tremendous capacities and cheap cost per bit of digital optical disks provides strong motivation to examine closely their one drawback – their "write-once" nature. The purpose of this paper is thus to explore the true capabilities of such "write-once memories" (or *woms*). Other familiar examples of woms are punched paper tape, punched cards, and PROMS (programmable read-only memories – in which the wits are microscopic fuses that can be selectively blown).

Large woms might naturally be used to store data that is more or less static: programs, documents, pictures, data bases, or archival storage dumps. If the data requires updating, the wom can be replaced by a freshly written wom. A large wom can be divided into blocks that are used up as needed; a index on an associated magnetic disk can keep track of the valid blocks. The magnetic disk can be eliminated by using linked-list or tree-like data structures on the wom itself to link obsolete blocks to their replacements, at some cost in terms of access time.

More formally, we model a wom as an array of "write-once bits" (or *wits*) which are manufactured in a "0" state but which can later be independently but irreversibly transformed into a "1" state. (We understand that some of the current recorder/player designs for digital optical disks are not capable of selectively changing individual zero bits within a previously written block. However, this seems to be more a matter of engineering than of fundamentals.)

The main result of this paper is that by using appropriate coding techniques, a wom can be "rewritten" many times, and that its "bit-capacity" is much greater than the number of its wits. Many of the coding techniques proposed here are simple to implement, and can have a significant impact on the cost of using woms.

As an example of the kind of behavior we are interested in, the following coding scheme was a prime "motivating example" for this research.

Lemma 1. Only 3 wits are needed to "write 2 bits twice".
Proof: We show how to represent a 2-bit value x in 3 wits so that it can later be changed to represent any other 2-bit value y. First, represent x with the pattern $r(x)$ given in Table 1. Later, a value y ($y \neq x$) can be written by changing the pattern to $r'(y)$. (If $x = y$ no change is made). Observe that $r'(y)$ will have ones wherever $r(x)$ does, so that we need only change zeros to ones.

x	$r(x)$	$r'(x)$
00	000	111
01	100	011
10	010	101
11	001	110

Table 1. A $\langle 2^2 \rangle^2/3$-Womcode

Decoding is easy: the memory word abc represents the 2-bit value $(b \oplus c)$, $(a \oplus c)$, no matter whether the wom has been written once or twice. ∎

II. Notation

Let *weight* of a binary codeword be the number of ones it contains. Let $x \oplus y$ denote the bitwise XOR of the bit vectors x and y (assumed to have the same length). We say that an binary word $x = x_1 \ldots x_r$ is "above" another binary word $y = y_1 \ldots y_s$ (denoted $x \geq y$) if $r = s$ and $x_i \geq y_i$ for $1 \leq i \leq r$. Let $\log(x)$ denote the logarithm (base 2) of x (or, if the context requires an integer value, $\lceil \log_2(x) \rceil$). We use Z_v to denote the set $\{0, 1, \ldots, v-1\}$, and Z_2^n to denote the set of all binary words of length n. We say "$f(n) \approx g(n)$" if $\lim_{n \to \infty} f(n)/g(n) = 1$. (The variable being taken to the limit should be clear from the context.) We also let $H(p)$ denote the "entropy function" $H(p) = p \log(1/p) + (1-p) \log(1/(1-p))$.

A coding scheme that uses n wits to represent one of v values so that it can be written a total of t times (i.e. written once and changed $t-1$ times) we call a "$\langle v \rangle^t/n$-womcode" (read: a "v t-times into n-wit womcode"). The "$/n$" may be dropped for an optimal wom code (with minimal n) (read: an "optimal v t-times womcode"). The general case – where the number of values may differ from generation to generation – we call a "$\langle v_1, \ldots, v_t \rangle/n$-womcode" (read: a "$v_1$ to v_t into n-wit womcode"); here the value stored on the i-th write may be any one of $\{0, \ldots, v_i - 1\}$.

Let $w(\langle v_1, \ldots, v_t \rangle)$ denote the least n for which a $\langle v_1, \ldots, v_t \rangle/n$-womcode exists. Similarly, we use $w(\langle v \rangle^t)$ to denote the number of wits needed by an optimal $\langle v \rangle^t$ womcode. We are interested in characterizing the behavior of $w(\langle v \rangle^t)$ for large values of v or t as well as finding "practical" coding schemes for small values of v or t.

It seems at first paradoxical that $w(\langle v \rangle^t) < \log(v) \cdot t$ could happen. Intuitively, the reason is that only the *last* value written needs to be accessible – previous values may

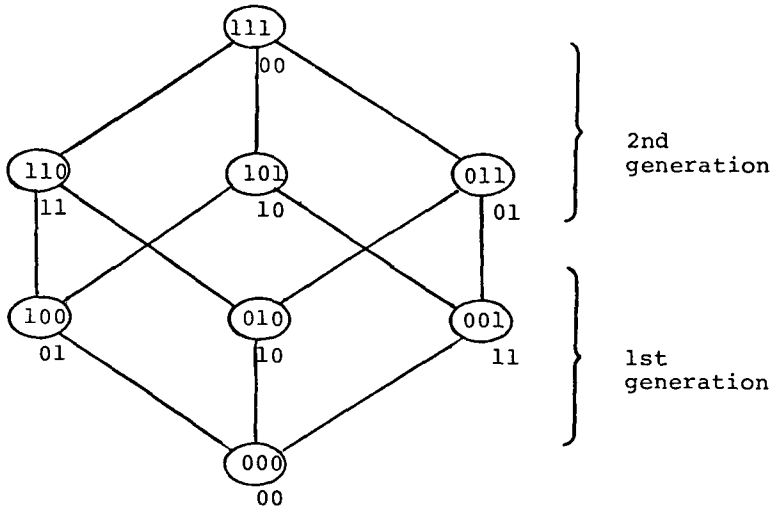

Figure 1. The $\langle 2^2 \rangle^2/3$-womcode on the Boolean 3-cube

become irretrievable. In fact, if all previously written values were always accessible then $\log(v)\cdot t$ wits would be required.

To make our definition of a womcode precise, we note that a $\langle v_1,\ldots,v_t\rangle/n$ womcode can be defined to consist of the following parts (here let $v=\max(v_1,\ldots,v_t)$):

(1) An *interpretation function* α that maps each $x\in Z_2^n$ to a *value* $\alpha(x)$ in Z_v,

(2) An *update function* μ which gives for any $x\in Z_2^n$ and for any value y in Z_v, either "$\perp$" (i.e. undefined), or else a codeword $\mu(x,y)=z\in Z_2^n$ such that $\alpha(z)=y$ and $z\geq x$.

We say that α and μ define a correct $\langle v_1,\ldots,v_t\rangle/n$-womcode if they "guarantee at least t writes" as defined below. In order to define this condition we first make the following auxiliary definitions.

An *acceptable sequence* $\langle i_1,\ldots,i_m\rangle$ for a $\langle v_1,\ldots,v_t\rangle/n$-womcode satisfies the conditions that $0\leq m\leq t$ and each i_j, $1\leq j\leq m$, is in Z_{v_j}. Note that in particular the null sequence λ is acceptable.

We define the "write function" ρ mapping acceptable sequences to codewords (or $\perp$) as the iteration of μ starting from the all-zero word 0^n (corresponding to the "initial state" of the wom) by the following equations:

$$\rho(\langle i_1,\ldots,i_j\rangle)=\begin{cases}0^n, & \text{if } j=0;\\ \mu(\rho(\langle i_1,\ldots,i_{j-1}\rangle),i_j), & \text{if } j\geq 1.\end{cases}$$

(We assume that $\mu(\perp,y)=\perp$ for all y.) We say that an acceptable sequence "arrives at x" if that sequence is mapped by ρ to x.

We say that a codeword x is "used in generation m," or "is an m-th generation codeword" if there is an acceptable sequence of length m that arrives at x. A codeword x is said to be "unused" if no acceptable sequence of positive length arrives at x, otherwise we say x is "used."

If every codeword belongs to at most one generation we call the womcode "synchronized" – since all acceptable sequences that arrive at a codeword word arrive there "at the same time" (i.e. at the same generation). Otherwise the womcode is called "unsynchronized." With a synchronized womcode one can always determine how many generations have been written. Note that our $\langle 2^2\rangle^2/3$ womcode is *not* synchronized since 000 belongs to the zeroth, first, and second generations. We say that a womcode is "almost synchronized" if the all-zero word is the only codeword that belongs to more than one generation, and it belongs only to the zeroth and first generations.

The *laminar* womcodes are an interesting special case of the synchronized womcodes: a womcode is *laminar* if it is synchronized and the weight of every (used) codeword determines its generation. (That is, no two codewords of different generations have the same weight.)

We say that the womcode defined by α and μ "guarantees at least t writes" if no acceptable sequence of length t arrives at $\perp$. This completes our formal definition of a $\langle v_1,\ldots,v_t\rangle/n$-womcode defined by α and μ; such a womcode is correct if it guarantees at least t writes.

We will often identify an interpretation $\alpha(x)$ with its binary representation. (For example, in a $\langle 2^k\rangle^t/n$-womcode each n-bit codeword represents a k-bit word.)

We would like to note that we initially studied only the $\langle 2^k\rangle^t/n$-womcodes, but that we have since seen enough interesting examples of womcodes of the more general form to warrant including the more general definition here.

We now introduce our three "complexity measures": P, I, and C.

Let $P(t)$ denote the "penalty expansion factor" needed to guarantee t writes of values from Z_v, for large v, compared to that needed for just a single write:

$$P(t)=\lim_{v\to\infty}\frac{w(\langle v\rangle^t)}{\log(v)}. \tag{1}$$

We will prove that $P(2)=1.29\ldots$ and $P(t)\approx t/\log(t)$.

Let $I(v)$ denote the asymptotic "incremental cost" of increasing t by one for a $\langle v\rangle^t$-womcode:

$$I(v)=\lim_{t\to\infty}\frac{w(\langle v\rangle^t)}{t} \tag{2}$$

We shall prove the surprising result that $I(v)=1$.

We define the apparent *capacity* (in bits) of a $\langle v_1,\ldots,v_t\rangle/n$-womcode to be $\log(v_1\cdots v_t)$; we denote this as $C(\langle v_1,\ldots,v_t\rangle/n)$. Similarly, let $C(n)$ denote the apparent *capacity* (in bits) of an n-wit memory field:

$$C(n)=\max\{\log(v_1\cdots v_t)\mid w(\langle v_1,\ldots,v_t\rangle)\leq n\}. \tag{3}$$

We shall demonstrate that $C(n)=n\cdot\log(n)+o(n\cdot\log(n))$. As an auxiliary definition, we let $R(\langle v_1,\ldots,v_t\rangle/n)=C(\langle v_1,\ldots,v_t\rangle/n)/n$ denote the *rate* of the womcode. (This is just the capacity per wit of the womcode.)

III. Elementary Observations

Lemma 2.

$$w(\langle v_1\cdot v_2\rangle^t)\leq w(\langle v_1\rangle^t)+w(\langle v_2\rangle^t) \tag{4}$$

Proof. Concatenate a $\langle v_1\rangle^t$-womcode and a $\langle v_2\rangle^t$-womcode to make a $\langle v_1\cdot v_2\rangle^t/(w(\langle v_1\rangle^t)+w(\langle v_2\rangle^t))$ womcode. (Represent each value y in $Z_{v_1\cdot v_2}$ as an ordered pair (y_1,y_2), with $y_1\in Z_{v_1}$ and $y_2\in Z_{v_2}$. Use the womcodes to record y_1 and y_2 separately.) ∎

Lemma 3.($w(\langle v\rangle^t)$ is subadditive in t.)

$$w(\langle v\rangle^{t_1+t_2})\leq w(\langle v\rangle^{t_1})+w(\langle v\rangle^{t_2}) \tag{5}$$

Proof. Use side-by-side optimal $\langle v\rangle^{t_1}$- and $\langle v\rangle^{t_2}$-womcodes to represent the sum (mod v) of the values represented by the two subcodes. To update, change one subcode to represent the difference (mod v) of the new value to be repre-

sented and the value of the other subcode. This guarantees at least $t_1 + t_2$ writes. (The alternative approach of writing the new value into one of the two subcodes would need extra wits to indicate which subcode was written last, unless the zero word is unused in one of the subcodes.) ∎

The above lemmas (and $w(\langle 2^1\rangle^1) = 1$) imply that $w(\langle 2^k\rangle^t) \leq k \cdot t$.

For small values of k and t, we can derive $w(\langle 2^k\rangle^t)$ as given in Table 2. (Obviously, $w(\langle 2^k\rangle^1) = k$ and $w(\langle 2^1\rangle^t) = t$.)

		t					
		1	2	3	4	5	6
	1	1	2	3	4	5	6
	2	2	3	5	6	7	
	3	3	5	7			
k	4	4	6				
	5	5	8				
	6	6	9				
	7	7					

Table 2. $W(\langle 2^k\rangle^t)$

We do not know the exact values corresponding to the empty positions of the table.

The $\langle 2^3\rangle^3/7$ (rate 1.28...)and $\langle 2^2\rangle^2/3$ (rate 1.33...) womcodes indicated by the table are special cases of the general "linear" scheme presented in section V.

The $\langle 2^2\rangle^5/7$ (rate 1.42...) womcode indicated by the table is an *ad hoc* scheme; we show here how to decode a 7-wit pattern *abcdefg*. If the pattern has weight four or less, the value represented is $01 \cdot c_{01} \oplus 10 \cdot c_{10} \oplus 11 \cdot c_{11}$, where $c_{01} = 1$ iff $ab = 10$ or ($ab = 11$ and one of *cd* or *ef* is 01), $c_{10} = 1$ iff $cd = 10$ or ($cd = 11$ and one of *ab* or *ef* is 01), and $c_{11} = 1$ iff $ef = 10$ or ($ef = 11$ and one of *ab* or *cd* is 01). (For example, the pattern 1101100 represents 10. At most one of *ab*, *cd*, *ef* will be 11 if another is 01.) Otherwise the interpretation is $ab \oplus cd \oplus ef \oplus gg$. The first three writes change at most one wit, while the last two might each change two wits.

The following notation for the size of the tail of a binomial distribution and a related inverse quantity will be useful:

$$\left(\!\!\binom{m}{h}\!\!\right) = \sum_{i=0}^{h} \binom{m}{i} \tag{6}$$

$$\delta(v, m) = \min\{\, h \mid \left(\!\!\binom{m+h}{h}\!\!\right) \geq v \,\}. \tag{7}$$

Note that a $\langle v\rangle^t/n$-womcode must have $n \geq m + \delta(v, m)$ if every first generation codeword must have at least m zeros. We derive a lower bound $Z(v, t)$ to $w(\langle v\rangle^t)$ by generalizing this observation:

$$Z(v, 0) = 0, \text{ and} \tag{8}$$

$$Z(v, t+1) = Z(v, t) + \delta(v, Z(v, t)) \text{ for } t \geq 0. \tag{9}$$

Lemma 4.

$$w(\langle v\rangle^t) \geq Z(v, t) \tag{10}$$

Proof. By induction on t. The case $t = 0$ is trivial. A $\langle v\rangle^{t+1}/n$-womcode must have at least $Z(v, t)$ zeros in every first-generation codeword, and must turn on at least $\delta(v, Z(v, t))$ wits in the worst case on the first write to have v codewords in the first generation. ∎

Corollary.

$$w(\langle 2^k\rangle^t) \geq k + t - 1 \tag{11}$$

Note that $Z(2^k, 1) = k$ and $Z(2^k, t+1) \geq Z(2^k, t) + 1$ for $k > 0$. The following lemma improves this result (by one).

Lemma 5.

$$w(\langle 2^k\rangle^t) \geq k + t \text{ for } k \geq 2 \text{ and } t \geq 3. \tag{12}$$

Proof. Suppose to the contrary that a $\langle 2^k\rangle^t/(k+t-1)$-womcode existed for $k \geq 2$ and $t \geq 3$. Since $Z(2^k, 1) = k$, the generation $t-1$ codewords must each have weight less than t. On the other hand, if $t \geq 3$ then for every value $y \in Z_{2^k}$ there is a $t-1$-st generation codeword x with $\alpha(x) = y$ and weight at least $t-1$. (For $t = 2$ the claim fails if the zero-weight word is in the first generation.) There must be at least $2^k - 1 \geq 3$ different values y associated with first-generation codewords of weight 1 or more. Thus for every value $y \in Z_{2^k}$ there is a $t-1$-st generation codeword x with $\alpha(x) = y$ and weight *exactly* $t-1$. But then no possible interpretation for the codeword 1^{k+t-1} is distinct from each of these values (required since the last k levels are "tight"). This contradiction proves the lemma. ∎

IV. How many wits are needed for a fixed number of generations?

IV.A. How many wits are needed for two generations?

Theorem 1.

$$w(\langle v\rangle^2) \approx 1.293815\ldots \log(v) \tag{13}$$

Proof. For any v, choose h to be $\delta(v, \log(v))$ and then choose n to satisfy:

$$n - h = \lceil \log(v) + \log\log(v) + 1 - \log\log(e) \rceil. \tag{14}$$

We will prove that $w(\langle v\rangle^2) \leq n$. Choose the first generation representations arbitrarily as distinct codewords with weight at most h, and randomly assign to the remaining $2^n - v$ codewords interpretations from Z_v. There are

$$(v)^{2^n - v} \tag{15}$$

ways to do this. How many ways do not guarantee two writes? Such a bad assignment must contain a first-generation codeword x and a value $y \in Z_v - \{\alpha(x)\}$ such that no codeword $z \geq x$ represents y. If we select x in one of v ways, select y in less than $v-1$ ways, assign all codewords $z \geq x$ values different than $\alpha(x)$ and assign all other codewords arbitrary values, we will have overcounted the bad codes but examined no more than

$$v^2 \cdot (v-1)^{2^{n-h}} \cdot (v)^{2^n - v - 2^{n-h}}. \tag{16}$$

codes. Whenever (16) is less than (15) some "good" codes must exist. This happens when

$$v^2 \leq \left(\frac{v}{v-1}\right)^{2^{n-h}} \tag{17}$$

which will happen if

$$2\log(v) \leq 2^{n-h-\log(v)} \cdot \log(e) \tag{18}$$

which is implied by

$$n = h + \lceil \log(v) + \log\log(v) + 1 - \log\log(e) \rceil. \tag{19}$$

Thus (19) implies the existence of a $\langle v \rangle^t / n$-womcode. Since $n \geq h + \log(v)$ (from Lemma 4), we conclude that for an optimal $\langle v \rangle^2 / n$-womcode

$$n = h + \log(v) + o(\log(v)). \tag{20}$$

Now the logarithm of the number of words of length n with at most h ones is

$$n \cdot H(h/n) + o(n) \text{ , for } h \leq n/2. \tag{21}$$

(See [PW72, Appendix A], or [MS77, Ch. 10, §11].) Since there are v values in the first generation,

$$n \cdot H(h/n) + o(n) = \log(v) \tag{22}$$

or (since $\log(v) = n - h + o(\log(v))$ and $n \leq 2 \cdot \log(v)$)

$$H(h/n) = \frac{(n-h)}{n} + o(1) \tag{23}$$

The equation $H(p) = 1 - p$ has a solution at $p = 0.22709219\ldots$, so for an optimal womcode $h/n \approx .227\ldots$, or $\log(v) \approx n \cdot (1 - .227\ldots)$ or

$$n \approx 1.29381537\ldots \cdot \log(v) \tag{24}$$

which was to be proved.∎

The random womcodes of the theorem will have an asymptotic rate of $2/1.29\ldots = 1.5458\ldots$, much better than the rate 1.33... womcode of lemma 1. However, we could not construct by hand a $\langle 2^k \rangle^2$-womcode of rate higher than 1.33.... Lemma 4 implies that such a scheme must have $k = 7, n = 10$ or $k \geq 9$. Using a computer we found a slightly more efficient method with rate 1.34....

The new scheme is a $\langle 26 \rangle^2/7$-womcode (rate $= 1.3429\ldots$). So a seven-track paper tape is "reusable" for writing just letters! Row i, column j of Table 3 gives the value (a letter) of the 7-bit string with binary value $i * 32 + j$. The first-generation is in upper case. Thus a "T" (0011000) is made into an "h" by changing bits 1, 2, and 5 (to obtain 1111100). We were unable to find a $\langle 27 \rangle^2/7$-womcode or to prove one doesn't exist, although we can prove that a $\langle 29 \rangle^2/7$ womcode doesn't exist.

```
  00000000001111111111222222222233
  01234567890123456789012345678901
 ---------------------------------
0 AHGGFYLwEZYrXfpnDWVzUdjoTwkeltdu
1 CSRcQiozPpihuexyOzsjsniwvcqgfkbm
2 BNMzLbgmKutbngfwJwrhkvxymjpsoqci
3 Ikmqlckuwteosdjvubdfgetpyxnlhrza
```

Table 3. A $(\langle 26 \rangle^2/7)$-womcode

IV.B. What is P(t)?

By reasoning similar to that of the proof of Theorem 1, we derived the following estimates for $P(t)$. Note how closely the estimates are to $t/log(t)$.

t	P(t)(est.)	t/log(t)
1	1.000	---
2	1.294	2.000
3	1.549	1.893
4	1.783	2.000
5	2.003	2.153
10	2.983	3.010
20	4.668	4.628
50	8.960	8.859
100	15.191	15.051
200	26.346	26.164

Table 4. $P(t)$(est.) vs. $t/\log(t)$

To demonstrate our main result that $P(t) \approx t/\log(t)$, we define an upper bound to $w(\langle v \rangle^t)$ which is asymptotically equal to our lower bound $Z(v,t)$ of Lemma 4, to within a small additive term.

Theorem 2. For fixed t and v sufficiently large, a sufficient condition for the existence of a $\langle v \rangle^t/n$-womcode is the existence of t numbers l_i, $1 \leq i \leq t$, such that

(a) $t \cdot \log(v) \geq n \geq l_1 \geq l_2 \geq \cdots \geq l_t \geq 0$,
(b) $\binom{n}{l_1} \geq v$,
(c) $\binom{l_i}{l_{i+1}} \geq v \cdot (t+1) \cdot \log(v)$, for $1 \leq i < t$.

Proof.

We prove the existence of a $\langle v \rangle^t/n$ womcode in which all i-th generation codewords contain exactly l_i *zeros*. Condition (b) implies that there are enough codewords with l_1

zeros for the v values of the first generation. We now show (by a counting argument) that for all i, $1 \leq i < t$ it is possible to assign interpretations to the codewords with l_{i+1} zeros for the $i+1$-st generation in such a way that for every j, $0 \leq j < v$, every codeword with l_i zeros is below some codeword with l_{i+1} zeros that has been assigned interpretation j.

The total number of ways in which the $\binom{n}{l_{i+1}}$ codewords with l_{i+1} zeros can assigned values is:

$$v^{\binom{n}{l_{i+1}}}. \tag{25}$$

We can overcount the number of "bad" ways of assigning interpretations to the codewords with l_{i+1} zeros (assuming we have already assigned interpretations to the earlier generations), in the following way. Choose a codeword x with l_i zeros, choose a "missing value" $y \in Z_v$, assign the $\binom{l_i}{l_{i+1}}$ codewords with l_{i+1} zeros above x with the $v-1$ remaining values (other than y), and assign the other codewords with l_{i+1} zeros arbitrary interpretations. The number of "bad" ways is thus at most:

$$\binom{n}{l_i} \cdot v \cdot (v-1)^{\binom{l_i}{l_{i+1}}} \cdot v^{\binom{n}{l_{i+1}} - \binom{l_i}{l_{i+1}}} \tag{26}$$

A "good" way must exist whenever (26) is less than (25). By simplifying this inequality, we get:

$$\binom{n}{l_i} \cdot v \cdot \left(1 - \frac{1}{v}\right)^{\binom{l_i}{l_{i+1}}} < 1. \tag{27}$$

Since $n \leq t \cdot \log(v)$ (otherwise the existence of the desired womcode is trivial), $\binom{n}{l_i} \leq v^t$, and thus it is enough to prove:

$$v^{t+1} \cdot \left(1 - \frac{1}{v}\right)^{\binom{l_i}{l_{i+1}}} < 1. \tag{28}$$

By condition (c), $\binom{l_i}{l_{i+1}} \geq v \cdot \log(v) \cdot (t+1)$, and thus it suffices to prove that:

$$v^{t+1} \cdot \left(1 - \frac{1}{v}\right)^{v \cdot \log(v) \cdot (t+1)} < 1. \tag{29}$$

But for large enough v, $(1 - \frac{1}{v})^v$ approaches $1/e$, and thus the left hand side is approximated by:

$$v^{t+1} \cdot e^{-\log(v) \cdot (t+1)}, \tag{30}$$

which approaches zero as v goes to infinity. ∎

To find the smallest (or nearly smallest) n for which the existence of a $\langle v \rangle^t$-womcode is guaranteed by the theorem, the numbers l_i should be chosen in reverse order (from l_t to l_1). The last two numbers can be chosen as:

$$l_t = \frac{log(v)}{2} + c \log\log(v), \tag{31}$$

where c is any constant greater than 1, and

$$l_{t-1} = \log(v) + 2c \log\log(v), \tag{32}$$

since

$$\binom{l_{t-1}}{l_t} = \binom{2l_t}{l_t} > \frac{2^{2l_t}}{2l_t} = \frac{v \cdot (\log(v)^{2c})}{\log(v) + 2c \log\log(v)}. \tag{33}$$

Since $c > 1$ and t is fixed, this becomes larger than $v \cdot \log(v) \cdot (t+1)$ for v sufficiently large. The other l_i's can chosen as the smallest numbers satisfying condition (c) of the theorem. Finally, n can be chosen as the smallest number satisfying $\binom{n}{l_1} \geq v$.

We now proceed to analyze the performance of the womcodes described above, in order to show that their performance is asymptotically equal to that of the lower bound we proved in Lemma 4. Then we prove our main theorem (theorem 4) that $P(t) \approx t/\log(t)$.

We first introduce some necessary notation. Let

$$\delta'(v,m) = \min\{\, h \mid \binom{m+h}{h} \geq v \,\}. \tag{34}$$

(Note the similarity to the definition of δ in (7).) Then we define:

$$Y_u(v,0) = \frac{\log(v)}{2} + c \log\log(v), \tag{35}$$

$$Y_u(v,1) = \log(v) + 2c \log\log(v), \text{ and} \tag{36}$$

$$Y_u(v,t+1) = Y_u(v,t) + \delta'(v \cdot u \cdot \log(v), Y_u(v,t)), \text{ for } t \geq 1. \tag{37}$$

For convenience in the next theorem, we define $Y(v,t)$ to be $Y_{\log(v)}(v,t)$; note that for large v, $Y(v,t) \geq Y_{t+1}(v,t)$. From this definition it follows that for v sufficiently large,

$$w(\langle v \rangle^t) \leq Y(v,t), \tag{38}$$

since $l_i = Y_{t+1}(v, t-i)$ for $1 \leq i \leq t$ and $n < Y_{t+1}(v,t)$ in the construction of last theorem.

Theorem 3. For $t \geq 1$,

$$\lim_{v \to \infty} \frac{Y(v,t)}{Z(v,t)} = 1. \tag{39}$$

Proof.

By induction on t. The case $t = 1$ is trivial. By comparing the forms of the definitions of Y and Z, we see that it is enough to prove:

$$\lim_{v \to \infty} \frac{\delta'(v \cdot (\log(v))^2, m')}{\delta(v,m)} = 1, \tag{40}$$

where $m = Z(v, t-1)$ and $m' = Y(v, t-1)$. We observe that

$$\delta(v,m) \leq \delta'(v,m) \leq \delta(v,m) + 1 \tag{41}$$

if $m \geq \log(v)$ (since that implies that $\delta(v,m) \leq (m/2)$). (Note that in (40) both m and m' are $\geq \log(v)$.) Thus we can replace δ' by δ in (40). Furthermore, $\delta(v,m)$ is a *decreasing* function of m, so that we can also replace m' by

the smaller value m in (40). In a similar vein, it is simple to show that $m \geq \log(v)$ implies that

$$\delta(v \cdot (\log(v))^2, m) \leq \delta(v, m) + 2\log\log(v), \qquad (42)$$

and a little more complicated to show that $\log(v) \leq m \leq \gamma \cdot \log(v)$ implies that

$$\delta(v, m) \geq \log(v) \cdot \left(2\gamma \cdot H^{-1}(1/2\gamma)\right) \qquad (43)$$

Combining these observations leads to the desired result. ∎

Theorem 4. $P(t) \approx t/\log(t)$.

Proof.

Let $n_t = Z(v, t)$. Then we must have:

$$v \leq \left(\!\!\binom{n_t}{n_t - n_{t-1}}\!\!\right) \approx 2^{n_t \cdot H(n_{t-1}/n_t)} \qquad (44)$$

so we derive

$$H(n_{t-1}/n_t) \approx \log(v)/n_t. \qquad (45)$$

We consider $H(p)$ near $p = 0$ using the fact that $H(p) = H(1-p)$:

$$1 - n_{t-1}/n_t \approx H^{-1}(\log(v)/n_t). \qquad (46)$$

Near $p = 0$, $H(p) \approx p \cdot \log(1/p)$, so $H^{-1}(y) \approx -y/\log(y)$:

$$1 - \frac{n_{t-1}}{n_t} \approx \frac{-\log(v)/n_t}{\log(\log(v)/n_t)} \qquad (47)$$

$$n_t - n_{t-1} \approx -\log(v)/\log(\log(v)/n_t) \qquad (48)$$

$$\frac{dn_t}{dt} \approx \frac{-\log(v)}{\log(\log(v)/n_t)} \qquad (49)$$

$$dt \approx \frac{log(n_t/\log(v)) \cdot dn_t}{\log(v)} \qquad (50)$$

$$t \approx (\frac{n_t}{\log(v)})\log(\frac{n_t}{\log(v)}) \qquad (51)$$

$$\frac{n_t}{\log(v)} \approx P(t) \approx \frac{t}{\log(t)} \qquad (52)$$ ∎

As a consequence of Theorem 4, for fixed t and large v, an optimal $\langle v \rangle^t$ womcode will have a rate approximately equal to $\log(t)$, with the approximation improving as t increases.

V. What is I(v)?

In this section we demonstrate that $I(v) = 1$ for any v, using a "tabular" womcode. We also present a "linear" womcode that – while it only shows that $I(v) \leq 4$, generalizes nicely our $\langle 2^2 \rangle^2/3$-womcode.

V.A. The Tabular $\langle v \rangle^t/n$-Womcode

We assume here that $t > v$. Let u denote an integer parameter to be chosen later (imagine that u is about $\log(n)$). Our $\langle v \rangle^t/n$-womcode will have its $n = r \cdot s$ wits considered as $r = (u+1)(v-1)$ rows of $s = \log(v) + t/(u \cdot (v-1))$ columns. Each row is divided into a $\log(v)$-wit "header" field and an $(s - \log(v))$-wit "count" field. The $\log(v)$-bit value represented by such a table is the sum (mod v)) of the header fields of all rows that have an odd number of "1"s in their count fields.

To write a value x when the table currently represents y, it suffices to change a single "0" to a "1" in a row with header $x - y \pmod v$. If every row with this header has all ones in its count field, we find a new row which currently is all zeros both its header and count fields, change the header to the desired value, and change one bit of the count field. We can always find a new row up until $u(v-1)$ rows are completely "full", since there are only $v-1$ useful header values. (The all-zero value is useless.) Thus we are guaranteed at least $u(v-1) \cdot (s - \log(v)) = t$ writes.

Since

$$n = t + t/u + \log(v)(u+1)(v-1),$$

by choosing $u = \lfloor \log(t) \rfloor$ implies that $n = t + o(t)$.

This code has rate approximately $\log(v) < \log(n)$. With optimally chosen parameters, this code has a rate nearly $\log(n)$, about twice as good as any other code presented in this paper.

V.B. The "Linear" Womcode

This scheme has parameters v, $t = 1 + v/4$, and $n = v - 1$. The i-th wit is associated with the number i, for $1 \leq i < v$. The value represented by any pattern is the sum (mod v) of the numbers associated with wits in the "1" state. (An alternative definition, useful when $v = 2^k$, interprets the pattern as the XOR of the k-bit representations of the numbers associated with wits in the "1" state. For example, in our $\langle 2^2 \rangle^2/3$-womcode the pattern abc can be decoded as $01 \cdot a \oplus 10 \cdot b \oplus 11 \cdot c$.)

We now show that – as long as there are at least $v/2$ zeros – we can change the wom to represent a new value by changing at most *two* wits. Let z denote the difference (modulo v) of the new value desired, y, and the current value represented, x. If the bit associated with z is now "0" we can simply change it to a "1". Otherwise let S denote the set of numbers associated with wits which are currently zero, and let T denote the set $\{ z - x \pmod v \mid x \in S \}$. Since $|S| = |T| \geq v/2$ and $|S \cup T| < v - 1$ (zero is in neither set), the set $S \cap T$ must be nonempty. Their overlap indicates a solution to the equation $z = x_1 + x_2 \pmod v$ where x_1 and x_2 are elements of S.

The code described above thus has rate roughly $\log(v)/4 \approx \log(n)/4$.

The linear womcode described above may have to stop when there are as many as $(v/2) - 1$ zeros left (which can happen after as few as $1 + (v/4)$ writes). The following trick allows one to keep going a while longer. Divide the n-wit field into $n/3$ blocks of size 3. By writing additional "1"s if necessary, force each block to have either one "1" or three "1"s exactly. Those "bad" blocks having no zeros remaining are now considered as deleted, while each of the remaining "good" blocks can be used to store one bit (by changing one of its wits – the other one is left untouched to indicate that the block is a good block and not a deleted block). With at least $(n-1)/2$ zeros remaining we are guaranteed of getting at least $n/12-1$ "good" blocks. The recurrence: $t(n) = n/4 + t(n/12)$ has the solution $t(n) = 3n/11 + O(1)$, indicating that this trick can increase the number of writes we can achieve by a factor of 12/11 (from $n/4$ to $3n/11$ writes). At the moment this coding trick is also the best general scheme we know of for making use of a "dirty" WOM that may have been written before in an arbitrary manner (i.e. without any thought of using some sort of "womcoding" for better utilization).

VI. What is C(n)?

The schemes presented in the last section can be used to show that $C(n) = n \cdot \log(n) + o(n \cdot \log(n))$.

Theorem 5.

$$C(n) \geq n \log(n) + o(n \log(n))$$

Proof. For a given large memory size n, we can use the "tabular" scheme of section V.A. and choose parameters: $\log(v) = \lfloor \log(n) - 2\log\log(n) \rfloor$ with $r = \lfloor \log\log(n) \rfloor \cdot (v-1)$ rows of length $s = \lfloor n/r \rfloor$. (We will "waste" $n - rs$ wits.) As before, the total number of writes possible is $n - o(n)$, proving the theorem. ∎

It is also possible to show that this result is "best possible":

Theorem 6.

$$C(n) \leq n \cdot \log(n)$$

Proof. (Intuitively, changing one wit out of n should provide at most $\log(n)$ bits of information.) Consider any $\langle v_1, \ldots, v_t \rangle / n$-womcode. The n-wit field can undergo at most $(t+1)^n < n^n$ "histories" as it progresses from its first state (all "0"s) to its final state (perhaps all "1"s), since we can describe the history by specifying for each of the n wits that it either always remains "0" or that it is turned to a "1" during one of the t write operations. On the other hand, the womcode has at least $v_1 \cdots v_t$ different acceptable sequences of length t to handle, each of which must have its own history. The theorem follows. ∎

VI. Other Womcodes

Several of our colleagues have become intrigued by the problem of designing high-rate womcodes, and have graciously consented to our sketching or referencing their preliminary results here.

- Prof. David Klarner (Dept. Math, SUNY Binghampton), has created an elegant $\langle 5 \rangle^3/5$ (rate 1.39...) *cyclic* womcode, which works as follows. The first value is represented by 10000 in the first generation, either 01001 or 00110 in the second generation, and one of 01111, 10110, or 11001 in the third generation. The other four values are handled similarly, using cyclic rotations of the words given for the first value. (Since n is prime all the cyclic rotations are distinct.)
- David Leavitt (an undergraduate working with Prof. Spyros Magliveras, Dept. Math, Univ. Nebraska at Lincoln), has found an even more efficient $\langle 7 \rangle^4/7$ (rate 1.60...) cyclic womcode by extending Klarner's technique. (To appear.)
- James B. Saxe (a graduate student at CMU) has created the following beautiful $\langle \frac{n}{2}, \frac{n}{2}-1, \ldots, 1 \rangle / n$ womcode (rate asymptotically $\frac{\log(n)}{2}$), where the two halves of each codeword are the same except that the left half has an extra "1" bit. The value represented is the number of zeros in the left half to the left of the extra bit. Each update (except the first), changes exactly two wits – one in each half.
- Saxe also suggested the following marvelous recursive womcode, which uses $n = 2^k$ wits, and changes exactly one wit per write. Using $f(n)$ to denote the capacity of Saxe's code, we shall see that $f(2^k) = k \cdot 2^{k-1}$, using the base case $f(1) = 0$, giving a rate of $\frac{\log(n)}{2}$. Partition the n wits into $\frac{n}{2}$ pairs. With the first $\frac{n}{2}$ writes we turn on at most one bit in each pair, and obtain capacity $f(\frac{n}{2}) + \frac{n}{2}$ by using the code recursively on the $\frac{n}{2}$ pairs, and getting an extra bit/write using the parity of the number of pairs whose left bit is on. For the second $\frac{n}{2}$ writes we obtain capacity $f(\frac{n}{2})$ recursively on the pairs by turning on their second bits as needed. The recurrence $f(n) = \frac{n}{2} + 2f(\frac{n}{2})$ gives the desired result.
- Saxe has also created a $\langle 65, 81, 63 \rangle / 12$ (rate 1.52...) womcode that can be improved to a $\langle 65, 81, 64 \rangle / 12$ (rate 1.53...) womcode if the "generation number" is externally available when decoding.

VII. Discussion and Conclusions

The results presented in this paper provide much information about the nature of the function $w(\langle v \rangle^t)$. On the basis of the evidence so far, we conjecture that

$$w(\langle v \rangle^t) \approx \max(t, \frac{\log(v) \cdot t}{\log(t)})$$

for all large v and t. We expect that this result should follow in a more-or-less straightforward manner from the results and techniques given here, but we have not as yet worked through a detailed demonstration. (Exercise for the reader: prove that $w(\langle 2^k \rangle^k) = \Theta(k^2/\log(k))$.)

The relationship between womcodes and error-correcting codes are interesting: we can view a womcode as a situation where the channel is assymmetric (only $0 \rightarrow 1$ errors occur), and where the transmitter knows where the errors will occur *before* he has to choose a codeword. Of course, there are still many differences, since the objective of womcoding is to allow many "messages" to be sent and the codeword for one message determines what the "errors" are for the next message.

A more closely related problem may be that of devising codes for random-access memories that have "stuck bits". Heegard [He81] has some recent work in this area. Again, however, the problem seems intrinsically different.

Some interesting work has been done on Turing machines that have "nonerasing" work tapes (e.g. see [Mi67]), which is peripherally related to the research reported here.

We note that our formulation of the problem requires that the decoding scheme for an $\langle v \rangle^t/n$-womcode provide a unique interpretation for each possible pattern of the n wits, independent of how many generations have been written. In some cases the current "generation number" might also be available as input to the decoding scheme at no extra cost (in wits). While this variation might permit some minor performance improvements in some instances, it remains an open question as to how much this additional information might help.

A number of questions need further investigation:

- What if there is some restriction on the kinds of updates that may occur? (For example, what if y can replace x only if y is numerically greater than x?)
- What advantages are there to representing a different number of values at each generation?
- What is the complexity of the decoding and updating algorithms for the best codes?
- How can these coding schemes be adapted to handle the possibility of errors occurring on the wom?
- If the underlying storage medium is viewed as storing a modulated digital signal rather than a sequence of bits, what kind of "womcoding" should be used to allow updating yet to maximize bandwidth while minimizing modulation frequency? (See [Br81], [HG69])
- What can be said about the average-case behavior of womcodes?
- What if the storage elements had more than two possible states, and had a complicated dag that described the set of legal state transitions?
- What truly practical womcodes exist?

Acknowledgements

We would like to thank Eric Brown, Abbas El Gamal, David Klarner, David Leavitt, Andrew Odlyzko, Michael Rabin, Jim Saxe, and Michael Sipser for their helpful comments and discussions.

References

[Br81] Brown, Eric Stewart. Digital Data Bases on Optical Videodisks. Bachelor of Science Thesis. (MIT, May 1981)

[Bu79] Bulthuls, K., M. Carasso, J. Heemskerk, P. Kivits, W. Kleuters, and P. Zalm, "Ten Billion Bits on a Disk" IEEE SPECTRUM (Aug. 1979), 26-33.

[Bu80] "Videodiscs: A Three-Way Race for a Billion-Dollar Jackpot" Business Week (July 7, 1980), 72-81.

[Ga68] Gallager, R. G. Information Theory and Reliable Communication. (Wiley 1968).

[Go82] Goldstein, C. M., "Optical Disk Technology and Information," *Science* **215**, No. 4534 (12 February 1982), 862-868.

[He81] Heegard, C. "Capacity and Coding for Computer Memory with Defects," Ph.D. Thesis, Stanford University Dept. of Statistics Technical Report No. 45, (May 1981).

[HG69] Hecht, M. and A. Guida, "Delay Modulation", Proc. IEEE, (Letters), (July 1969).

[MS77] MacWilliams, F. J., and N. J. A. Sloane, The Theory of Error-Correcting Codes (North-Holland, Amsterdam, 1977).

[Mc81] McLeod, J. "Optical Disks loom as replacement for Tape" Electronic Design (Sept. 30, 1981), 97-103.

[Mi67] Minsky, M. Computation: Finite and Infinite Machines. (Prentice-Hall, 1967).

[PW72] Peterson, W. and E. Weldon Jr. Error-Correcting Codes. (MIT Press, 1972).