Constrained Codes for Multilevel Flash Memory

Paul H. Siegel

Center for Magnetic Recording Research
University of California, San Diego

North American School
of Information Theory

August 12, 2015

Constrained Codes for Multilevel Flash Memory

Paul H. Siegel

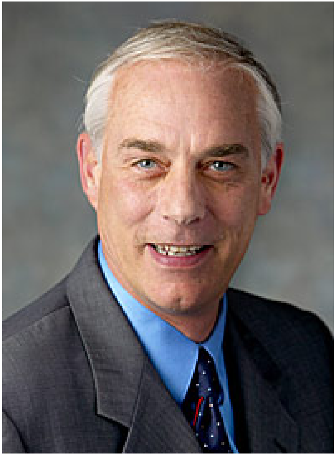
Center for Memory and Recording Research
University of California, San Diego

CMRR

North American School
of Information Theory

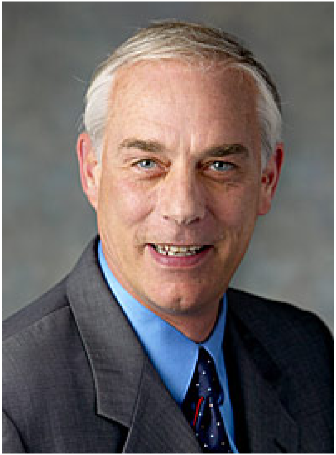
August 12, 2015

Acknowledgement



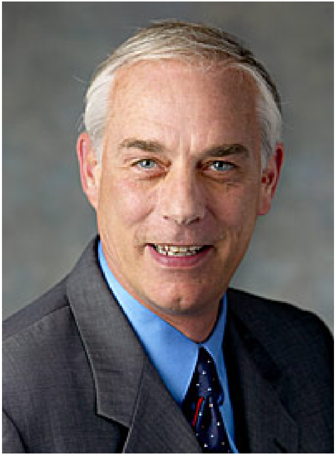
Dr. Roberto Padovani

Acknowledgement



Dr. Roberto Padovani

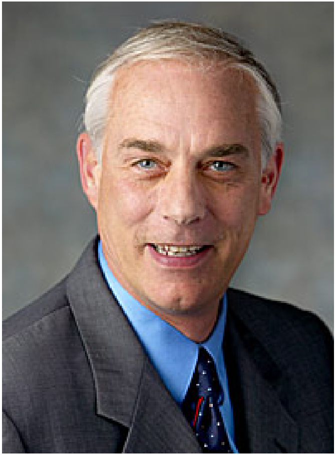
Thanks, Roberto ...



Dr. Roberto Padovani

Thanks, Roberto ...

for endowing this lectureship



Dr. Roberto Padovani

Thanks, Roberto ...

for endowing this lectureship

and for

CDMA-based mobile phones!

To our friend and colleague ...

Jack Keil Wolf - 2010 Padovani Lecturer

Introduction

- The theory of constrained coding began with Claude Shannon's 1948 paper, "A Mathematical Theory of Communication."
- In the Introduction, he presented the model of a general communication system, in the celebrated "Fig. 1":

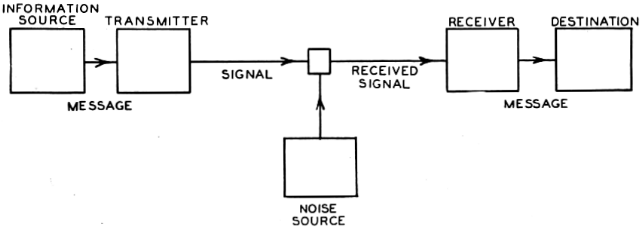


Fig. 1—Schematic diagram of a general communication system.

- In Part I, Section 1, he defined a discrete noiseless channel: a system allowing transmission of a set of finite sequences over an alphabet, subject to certain constraints.
- We'll call such a channel a constrained channel.
- His example – the telegraph channel, in "Fig. 2":

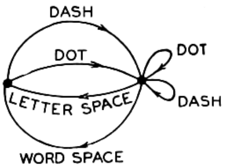
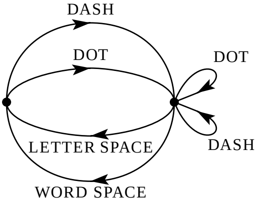


Fig. 2—Graphical representation of the constraints on telegraph symbols.

Telegraph channel

- The telegraph channel allows certain constrained sequences of symbols denoted DOT, DASH, LETTER SPACE, and WORD SPACE.



- The symbols have duration 2, 4, 3, and 6 time units, represented by 10, 1110, 000, and 000000.

- The constraint is that no spaces can follow each other. (Note that two letter spaces equal a word space.)

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

  Q2: Do there exist coding algorithms that efficiently translate information sequences into allowable sequences?

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

  Q2: Do there exist coding algorithms that efficiently translate information sequences into allowable sequences?

- His answers:

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

  Q2: Do there exist coding algorithms that efficiently translate information sequences into allowable sequences?

- His answers:

  A1: The asymptotic growth rate C of the number of allowable signals of duration T time units.

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

  Q2: Do there exist coding algorithms that efficiently translate information sequences into allowable sequences?

- His answers:

  A1: The asymptotic growth rate C of the number of allowable signals of duration T time units.
  (For the telegraph channel, $C = 0.539$ bits per time unit.)

Constrained coding

- Shannon asked two general questions about this and other constrained channels:

  Q1: How do we measure the capacity of the channel to transmit information (in bits per unit time)?

  Q2: Do there exist coding algorithms that efficiently translate information sequences into allowable sequences?

- His answers:

  A1: The asymptotic growth rate C of the number of allowable signals of duration T time units.
  (For the telegraph channel, $C = 0.539$ bits per time unit.)

  A2: Yes.

Fundamental Theorem for Noiseless Channel

Theorem (Shannon, 1948)

Let a source have entropy H (bits per symbol) and a channel have a capacity C (bits per time unit). Then it is possible to encode the output of the source in such a way as to transmit at the average rate $\frac{C}{H} - \epsilon$ symbols per time unit over the channel where ϵ is arbitrarily small. It is not possible to transmit at an average rate greater than $\frac{C}{H}$.

- The proof is non-constructive (typical sequences).
- If the source is binary and unconstrained, then $H = 1$, and achievable transmission rates approach the channel capacity C.

Morse code

- The Morse code is a combined source-constrained code for the English language over the telegraph channel.

A	•–	J	•–––	S	•••	1	•––––
B	–•••	K	–•–	T	–	2	••–––
C	–•–•	L	•–••	U	••–	3	•••––
D	–••	M	––	V	•••–	4	••••–
E	•	N	–•	W	•––	5	•••••
F	••–•	O	–––	X	–••–	6	–••••
G	––•	P	•––•	Y	–•––	7	––•••
H	••••	Q	––•–	Z	––••	8	–––••
I	••	R	•–•	0	–––––	9	––––•

- The Morse code is a combined source-constrained code for the English language over the telegraph channel.

A	·–	J	·–––	S	···	1	·––––
B	–···	K	–·–	T	–	2	··–––
C	–·–·	L	·–··	U	··–	3	···––
D	–··	M	––	V	···–	4	····–
E	·	N	–·	W	·––	5	·····
F	··–·	O	–––	X	–··–	6	–····
G	––·	P	·––·	Y	–·––	7	––···
H	····	Q	––·–	Z	––··	8	–––··
I	··	R	·–·	0	–––––	9	––––·

- The last telegram was sent on July 14, 2013.

The Morse code lives on...

The Morse code lives on...

IC School at EPFL

The Morse code lives on...

Sylvain Froidevaux - SCENICVIEW

IC School at EPFL (Building BC)

A closer look ...

IC School at EPFL (Building BC)

Decoding the IC School

.. -. ..-. --- .-. -- .- - .. --.- ..- .   . -   -.-. --- -- -- ..- -. .. -.-. .- - .. --- -. ...

i n f o r m a t i q u e e t c o m m u n i c a t i o n s

A	•–	J	•–––	S	•••	1	•––––
B	–•••	K	–•–	T	–	2	••–––
C	–•–•	L	•–••	U	••–	3	•••––
D	–••	M	––	V	•••–	4	••••–
E	•	N	–•	W	•––	5	•••••
F	••–•	O	–––	X	–••–	6	–••••
G	––•	P	•––•	Y	–•––	7	––•••
H	••••	Q	––•–	Z	––••	8	–––••
I	••	R	•–•	0	–––––	9	––––•

Constrained coding

- As with channel coding and source coding, Shannon's results launched a new field of research: coding for constrained channels.

- Since the 1960s, data storage technology has consistently spurred progress in the theory and design of constrained codes, and *vice versa*.

- New fundamental problems, deep mathematical results, practical code design techniques, and connections to other disciplines have been – and continue to be – found.

- This lecture will describe a selection of these developments in the context of constrained coding for multilevel flash memory.

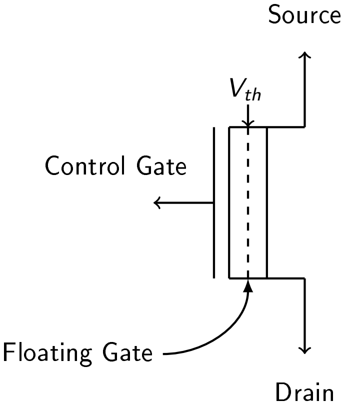
Outline

- Flash memory basics
- One-dimensional (1D) constrained codes
- Two-dimensional (2D) constrained codes
- Concluding remarks

Flash Memory Basics

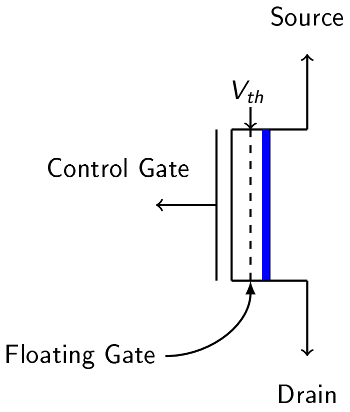
Flash memory cell

- Floating gate transistor: the basic flash memory unit (cell).
- Program via charge injection: threshold voltage represents stored bit values



- Incremental Step Pulse Program (ISPP)
- Increasing cell level is easy.
- Decreasing cell level is hard (more on this later)

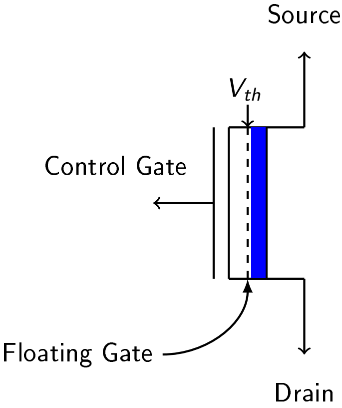
- Floating gate transistor: the basic flash memory unit (cell).
- Program via charge injection: threshold voltage represents stored bit values



- Incremental Step Pulse Program (ISPP)
- Increasing cell level is easy.
- Decreasing cell level is hard (more on this later)

Flash memory cell

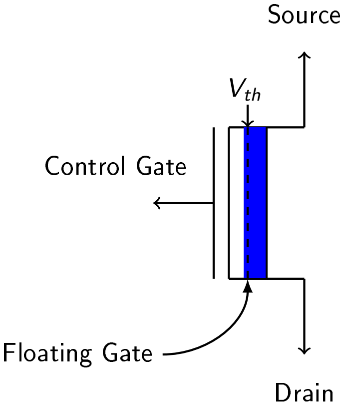
- Floating gate transistor: the basic flash memory unit (cell).
- Program via charge injection: threshold voltage represents stored bit values



- Incremental Step Pulse Program (ISPP)
- Increasing cell level is easy.
- Decreasing cell level is hard (more on this later)

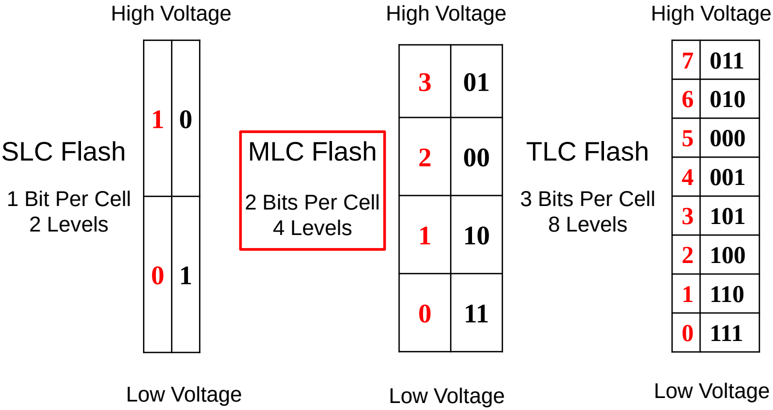
Flash memory cell

- Floating gate transistor: the basic flash memory unit (cell).
- Program via charge injection: threshold voltage represents stored bit values



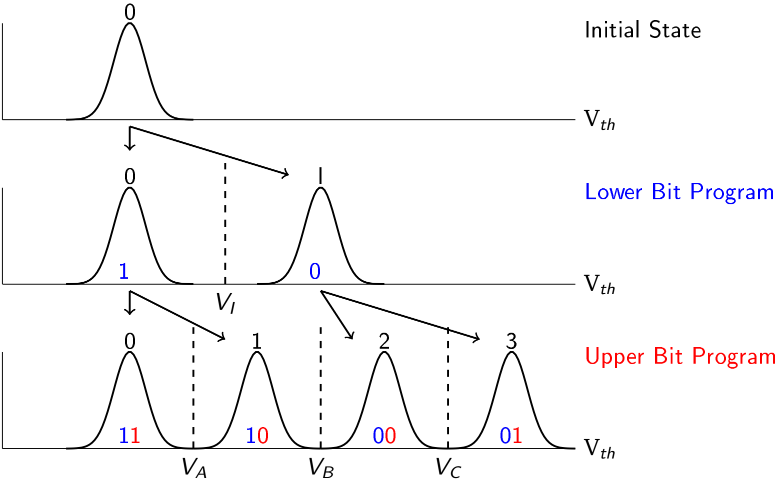
- Incremental Step Pulse Program (ISPP)
- Increasing cell level is easy.
- Decreasing cell level is hard (more on this later)

Common types of flash memory

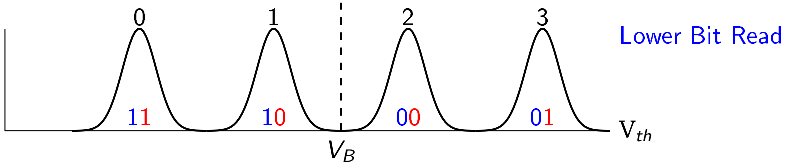


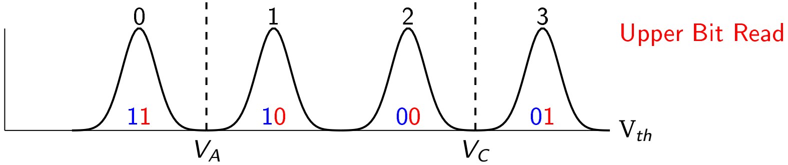
- Binary patterns are assigned to cell levels using a Gray code.
- In MLC flash, the two bits are called the lower and upper bits.

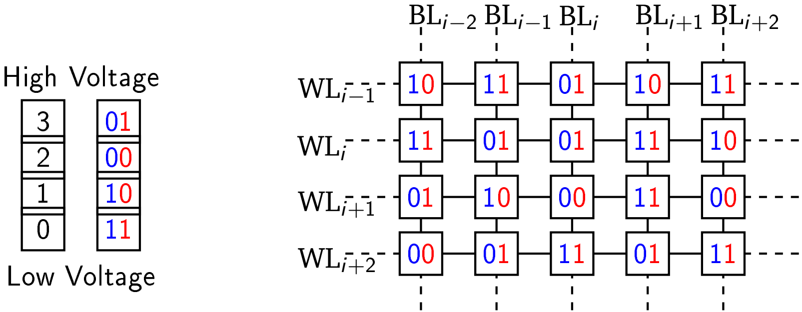
Programming MLC flash cells



Reading MLC flash cells

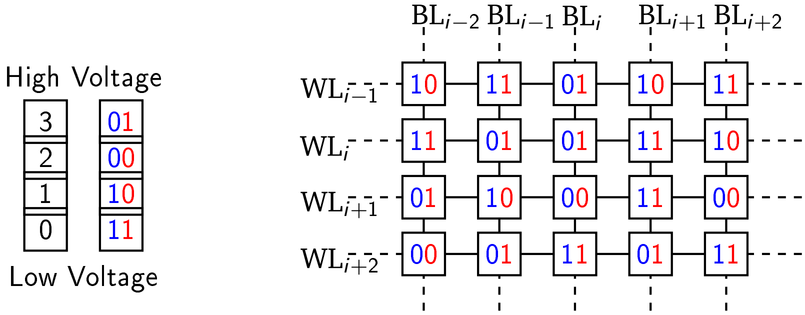






- Cells are arranged in an array, called a block.
- Rows (wordlines) are ~128K cells; columns (bitlines) are ~64 cells.
- In each wordline, lower bits of cells constitute the lower page, and upper bits constitute the upper page.

MLC flash memory structure

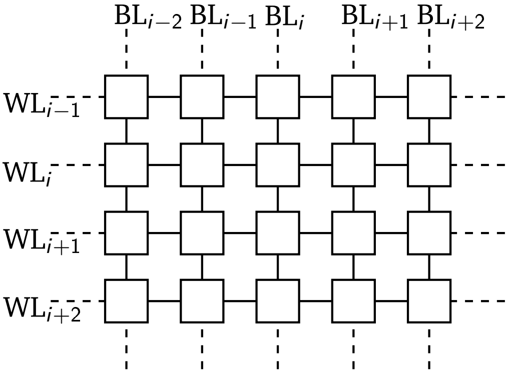


- Pages are the basic unit for read and write operations.
- Once programmed, a page can be rewritten only after the entire containing block is erased.
- Block erasures cause damaging wear on the flash memory cells, and are to be avoided.

Programming MLC flash blocks

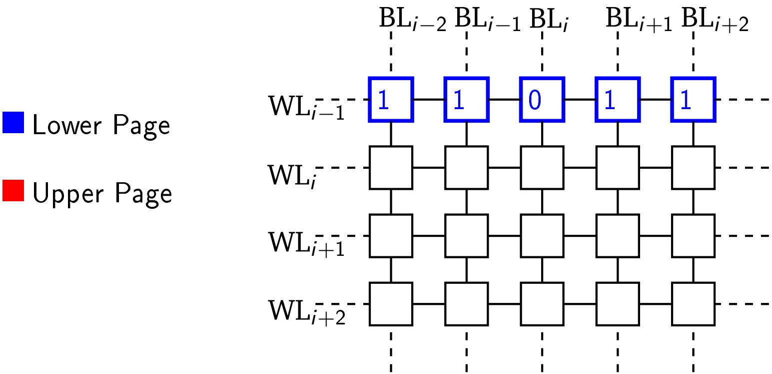
Lower Page

Upper Page



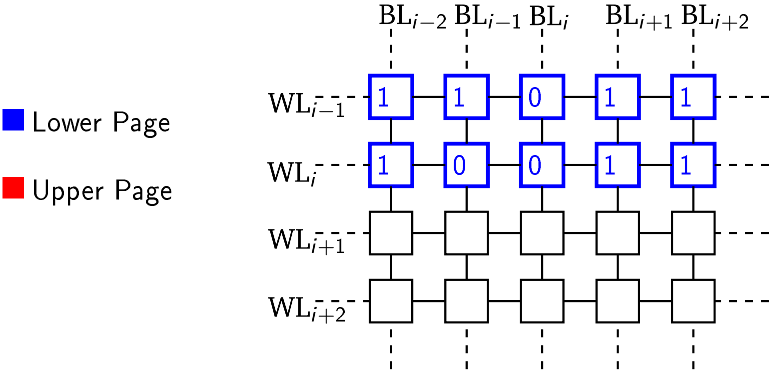
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



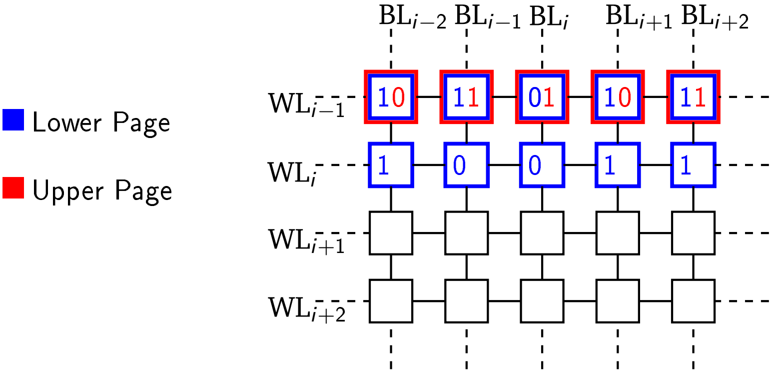
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



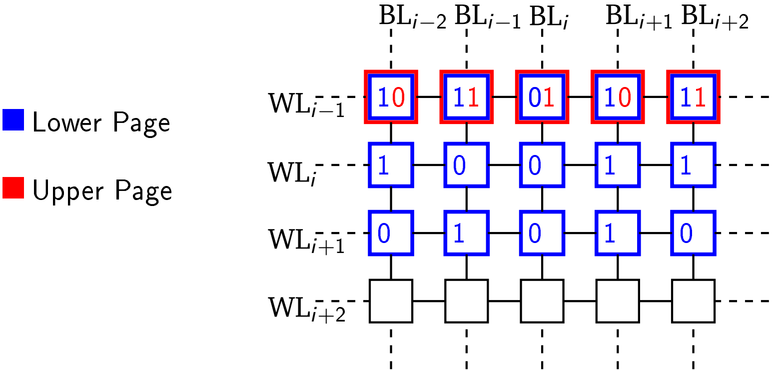
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



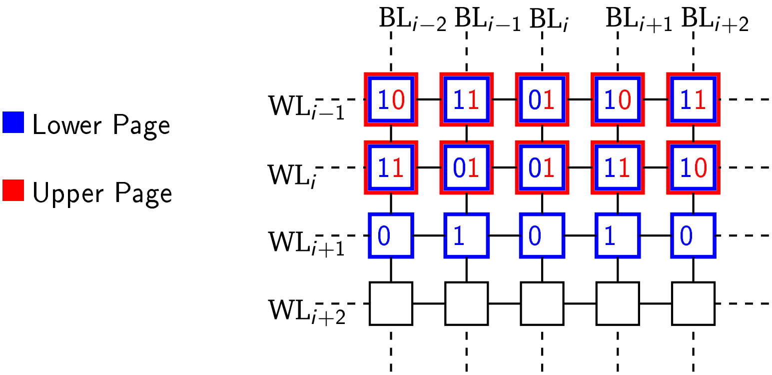
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



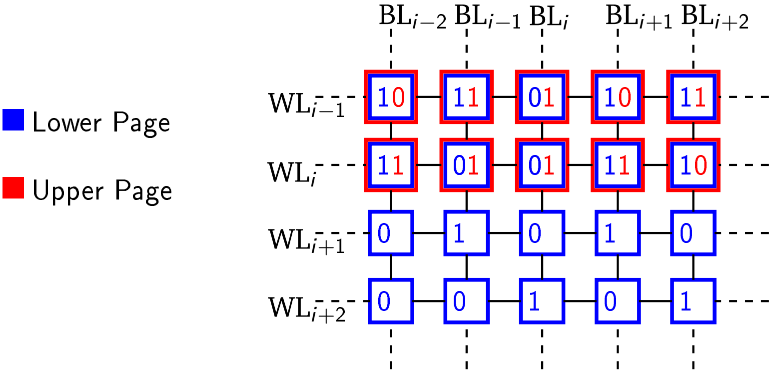
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



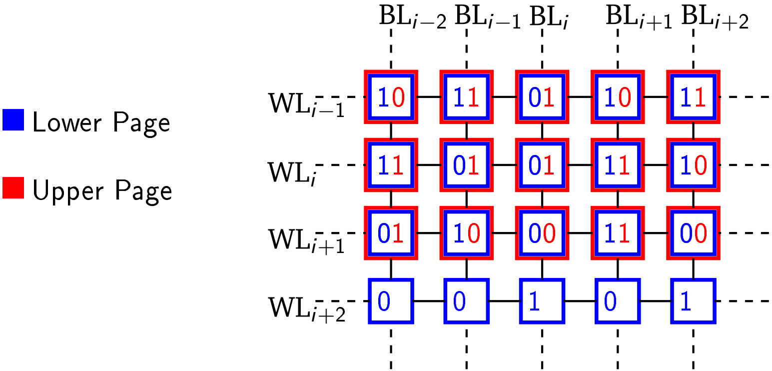
- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

Programming MLC flash blocks



- Upper and lower pages are independent.
- Pages are programmed row-by-row in a sequential order.

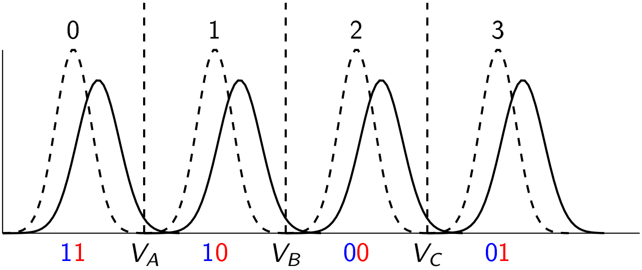
Error mechanisms in flash memories

- Program/Erase (P/E) cycling
  - Block erasures cause cell wear
  - Affects lifetime and reliability.
- Inter-cell Interference (ICI)
  - Cell coupling leads to data-dependent errors after programming.
  - Affects reliability.
- Charge loss over time
  - Programmed charge decays over time.
  - Affects data retention.

Error mechanisms in flash memories

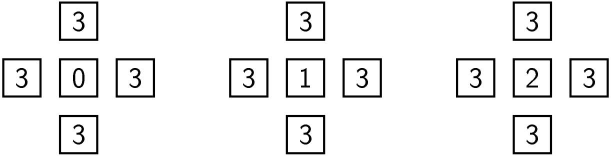
- Program/Erase (P/E) cycling
  - Block erasures cause cell wear
  - Affects lifetime and reliability.
- Inter-cell Interference (ICI)
  - Cell coupling leads to data-dependent errors after programming.
  - Affects reliability.
- Charge loss over time
  - Programmed charge decays over time.
  - Affects data retention.

- 96.5% of cell errors are adjacent cell-level errors in the upward direction, caused by inter-cell interference (ICI).



$0 \to 1$ (upper page error)
$1 \to 2$ (lower page error)
$2 \to 3$ (upper page error)

Dominant cell error patterns



- Neighbor cells programmed to level '3' cause the most ICI.
- Worst-case patterns are 3-0-3, 3-1-3, and 3-2-3 along wordlines, bitlines, or both.
- Bitline ICI induces more errors.

ICI-mitigation via coding

- How can we use coding to reduce the impact of ICI-induced errors in flash memory?
- One way is to use an error correcting code, such as a BCH or LDPC code, applied independently to every page.
- This is what is done today.

- Another way currently being explored is to ensure that the ICI-prone cell-level patterns along wordlines and bitlines are never programmed into the memory in the first place.
- This is where constrained coding can help.
- Let's see how we can apply it to MLC flash memory...

Outline (elaborated)

- Flash memory basics
- One-dimensional (1D) constrained codes
  - Wordline page coding
  - Joint wordline page coding
- Two-dimensional (2D) constrained codes
  - Row-by-row bitline coding
  - Combined wordline and bitline coding
- Concluding remarks

1D: Wordline Page Coding

Error-prone cell patterns - binary representations

- Under the restriction of independent programming of wordline pages, how can we eliminate error-prone cell-level patterns?

- Under the restriction of independent programming of wordline pages, how can we eliminate error-prone cell-level patterns?

- Consider the binary representation of the most susceptible patterns: 3-0-3, 3-1-3, 3-2-3.

Cell levels	3-0-3	3-1-3	3-2-3
U	1 1 1	1 0 1	1 0 1
L	0 1 0	0 1 0	0 0 0

Error-prone cell patterns - binary representations

- Under the restriction of independent programming of wordline pages, how can we eliminate error-prone cell-level patterns?

- Consider the binary representation of the most susceptible patterns: 3-0-3, 3-1-3, 3-2-3.

Cell levels	3-0-3	3-1-3	3-2-3
U	1 1 1	1 0 1	1 0 1
L	0 1 0	0 1 0	0 0 0

- We can forbid the strings 111 and 101 in wordline upper pages; i.e., a “no 1X1” constraint.

Error-prone cell patterns - binary representations

- Under the restriction of independent programming of wordline pages, how can we eliminate error-prone cell-level patterns?

- Consider the binary representation of the most susceptible patterns: 3-0-3, 3-1-3, 3-2-3.

Cell levels	3-0-3	3-1-3	3-2-3
U	1 1 1	1 0 1	1 0 1
L	0 1 0	0 1 0	0 0 0

- We can forbid the strings 111 and 101 in wordline upper pages; i.e., a "no 1X1" constraint.

- Equivalently, we can forbid 000 and 010 in wordline lower pages; i.e., a "no 0X0" constraint.

Error-prone cell patterns - binary representations

- Under the restriction of independent programming of wordline pages, how can we eliminate error-prone cell-level patterns?

- Consider the binary representation of the most susceptible patterns: 3-0-3, 3-1-3, 3-2-3.

Cell levels	3-0-3	3-1-3	3-2-3	3-3-3
U	1 1 1	1 0 1	1 0 1	1 1 1
L	0 1 0	0 1 0	0 0 0	0 0 0

- We can forbid the strings 111 and 101 in wordline upper pages; i.e., a "no 1X1" constraint.

- Equivalently, we can forbid 000 and 010 in wordline lower pages; i.e., a "no 0X0" constraint.

No 00 constraint

- We will impose a “no 0X0” constraint on lower pages: no adjacent 0’s in even positions or in odd positions.
- On interleaved subpages, this becomes a “no 00” contraint.

No 00 constraint

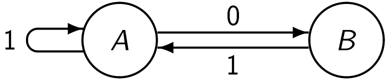
- We will impose a "no 0X0" constraint on lower pages: no adjacent 0's in even positions or in odd positions.

- On interleaved subpages, this becomes a "no 00" contraint.

- The "no 00" constraint means that 0s are isolated, e.g.,

  0 1 0 1 1 1 0 1 1 .

- As with the telegraph constraint, we can describe the allowable words of the "no 00" constraint in terms of edge labelings of paths on a directed graph:



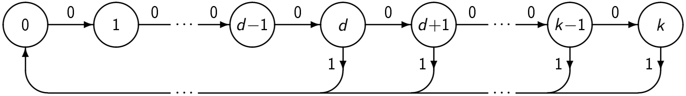
Constrained systems

- A labeled graph $G = (V, E, L)$ consists of:
  - a finite set of vertices, or states, V
  - a finite set of directed edges, E, with initial and terminal states in V
  - a labeling function on edges, $L : E \to \Sigma$, where Σ is finite alphabet.
- We assume G is lossless: distinct paths with the same initial state and terminal state have different labelings.
- A constrained system, denoted S, is the set of words obtained by reading the edge labels of finite paths in a labeled, directed graph G. We write $S = S(G)$.

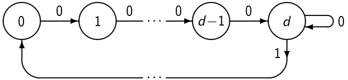
Runlength-limited (RLL) constraints

- The (d, k)-RLL constraint, $S_{d,k}$, contains all binary words with runlengths of 0s no more than k, and at least d between consecutive 1s.

- RLL constraints are used in magnetic and optical recording.



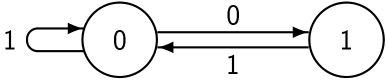
(d, k) constraint, k finite



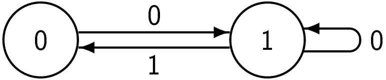
(d, ∞) constraint

$(0,1)$ and $(1,\infty)$ RLL constraints

- The "no 00" constraint is the $(0,1)$-RLL constraint.



- The "no 11" constraint is the $(1,\infty)$-RLL constraint.



- These constraints are bit-wise complements of one another.

- The capacity of a constrained system S, denoted $\text{cap}(S)$, is defined by
$$\text{cap}(S) = \limsup_{n \to \infty} \frac{1}{n} \log_2 N(n; S)$$
where
$$N(n; S) \text{ is the number of words of length } n \text{ in } S.$$

Combinatorial characterization of capacity

- The capacity of a constrained system S, denoted $\text{cap}(S)$, is defined by

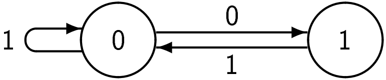
$$\text{cap}(S) = \limsup_{n \to \infty} \frac{1}{n} \log_2 N(n; S)$$

where

$N(n; S)$ is the number of words of length n in S.

- The 'lim sup' can be replaced by a 'lim' by subadditivity.
- Capacity measures the growth rate of the number of sequences of length n, i.e., $N(n; S) \approx 2^{n\,\text{cap}(S)}$.

Computation of cap($S_{0,1}$)

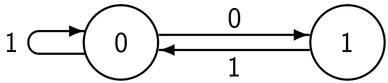


- Number of words $N_0(n)$ generated from state 0:

$$N_0(n+2) = N_0(n+1) + N_0(n), \ \forall\ n \geq 0$$

with $N_0(0) = 1$ and $N_0(1) = 2$.

Computation of $\mathrm{cap}(S_{0,1})$

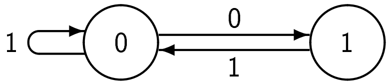


- Number of words $N_0(n)$ generated from state 0:

$$N_0(n+2) = N_0(n+1) + N_0(n),\ \forall\ n \geq 0$$

with $N_0(0) = 1$ and $N_0(1) = 2$.

- $N_0(n)$ is Fibonacci number f_{n+2}, with $f_n = \frac{1}{\sqrt{5}}\left[\lambda^n - (-\lambda)^{-n}\right]$, where $\lambda = \frac{1+\sqrt{5}}{2}$, the largest real root of $x^2 - x - 1$.



- Number of words $N_0(n)$ generated from state 0:

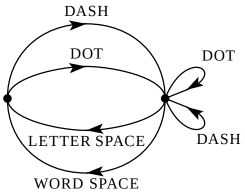
$$N_0(n+2) = N_0(n+1) + N_0(n), \ \forall\ n \geq 0$$

with $N_0(0) = 1$ and $N_0(1) = 2$.

- $N_0(n)$ is Fibonacci number f_{n+2}, with $f_n = \frac{1}{\sqrt{5}}\left[\lambda^n - (-\lambda)^{-n}\right]$, where $\lambda = \frac{1+\sqrt{5}}{2}$, the largest real root of $x^2 - x - 1$.

- So,

$$\text{cap}(S_{0,1}) = \lim_{n\to\infty} \frac{\log_2(f_{n+2})}{n} = \log_2(\lambda) \approx 0.6942$$



Symbol	Duration
DOT	2
DASH	4
LETTER	3
WORD	6

- The difference equation is:

$$N(n) = N(n-2)+N(n-4)+N(n-5)+N(n-7)+N(n-8)+N(n-10)$$

- $N(n)$ grows like $c\lambda^n$, where λ is the largest real root of

$$1-(x^{-2}+x^{-4}+x^{-5}+x^{-7}+x^{-8}+x^{-10}).$$

- Therefore, $\text{cap}(S_{\text{telegraph}}) = \log_2(\lambda) \approx 0.5389$.

Algebraic characterization of capacity

- We can compute capacity using the adjacency matrix A_G:

$$A_G = \left[(A_G)_{(u,v)}\right], \; u, v \in V$$

where $(A_G)_{(u,v)}$ is the number of edges from u to v.

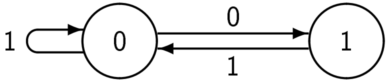
Algebraic characterization of capacity

- We can compute capacity using the adjacency matrix A_G:

$$A_G = \left[(A_G)_{(u,v)} \right], \; u, v \in V$$

where $(A_G)_{(u,v)}$ is the number of edges from u to v.

- For the $(0,1)$-RLL graph, $A_G = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.



Algebraic computation of cap(S)

Theorem (Shannon, 1948)

Let G be an irreducible, lossless presentation of S. Then,

$$\text{cap}(S) = \log_2 \lambda(A_G)$$

where $\lambda(A_G)$ is the largest real eigenvalue of A_G.

- A graph G is irreducible if for any ordered pair of states u, v there is a path from u to v.
- For $(0,1)$-RLL, we have $\lambda(A_G) = \frac{1+\sqrt{5}}{2}$, so

$$\text{cap}(S) = \log_2 \frac{1+\sqrt{5}}{2} \approx 0.6942.$$

Computing cap(S) (variable-length labels)

Theorem (Shannon, 1948)

Let b_{ij}^s be the duration of the s^{th} symbol which is allowable in state i and leads to state j. Then the channel capacity C is equal to $\log \lambda$ where λ is the largest real root of the determinental equation:

$$\left| \Sigma_s \lambda^{-b_{ij}^s} - \delta_{ij} \right| = 0$$

where $\delta_{ij} = 1$ if $i = j$ and is zero otherwise.

- For the telegraph channel, the equation is

$$\begin{vmatrix} -1 & (\lambda^{-2} + \lambda^{-4}) \\ (\lambda^{-3} + \lambda^{-6}) & (\lambda^{-2} + \lambda^{-4} - 1) \end{vmatrix} = 0.$$

Capacity formulas

- For $0 \leq d < k < \infty$, $C(d,k) \stackrel{\text{def}}{=} \text{cap}(S_{d,k}) = \log_2(\lambda_{d,k})$, where $\lambda_{d,k}$ is the largest real solution of the equation

$$x^{k+1} - x^{k-d} - \ldots - x - 1 = 0.$$

- For $d > 0$, $C(d,\infty) \stackrel{\text{def}}{=} \text{cap}(S_{d,\infty}) = \log_2(\lambda_{d,\infty})$, where $\lambda_{d,\infty}$ is the largest real solution of the equation

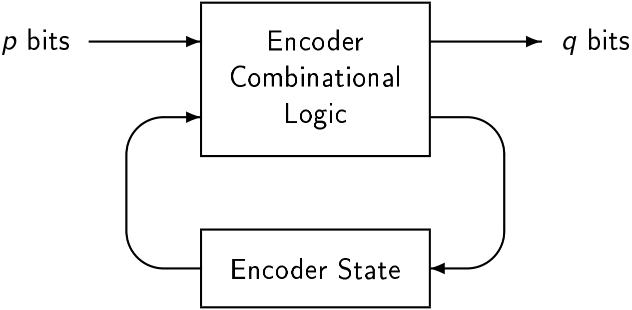
$$x^{d+1} - x^{d} - 1 = 0.$$

- Some (d,k)-RLL capacities

$k \backslash d$	0	1	2	3	4	5
1	.6942					
2	.8791	.4057				
3	.9468	.5515	.2878			
4	.9752	.6174	.4057	.2232		
5	.9881	.6509	.4650	.3218	.1823	
6	.9942	.6690	.4979	.3746	.2669	.1542
7	.9971	.6793	.5174	.4057	.3142	.2281
∞	1.0000	.6942	.5515	.4650	.4057	.3620

- These are all irrational except for $C(0,\infty)$.

- Practical encoders are fixed-rate, finite-state-machines.



- If the encoder has only one state, it is a block encoder, i.e., a look-up table.

Shannon's Coding Theorem

Theorem (Converse to coding theorem)

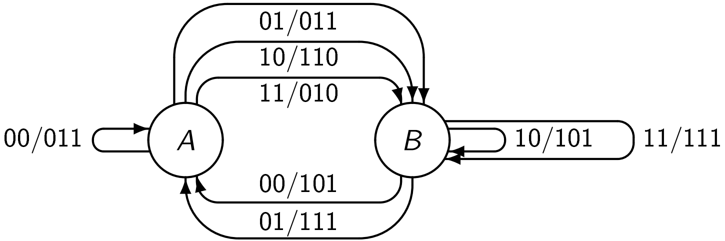
If there exists a rate $p : q$ encoder for S, then $p/q \leq \text{cap}(S)$.

Theorem (Block coding theorem)

There exists a sequence of rate $p_m : q_m$ block encoders for S such that $\lim_{m \to \infty} p_m/q_m = \text{cap}(S)$.

- Minimum block sizes for rates near capacity may be large.
  - $C(0,1) \approx 0.6942$: rate $2/3$, $p : q = 12 : 18$.
  - $C(1,7) \approx 0.6793$: rate $2/3$, $p : q = 42 : 63$.
  - $C(2,7) \approx 0.5174$: rate $1/2$, $p : q = 17 : 34$.

Rate 2 : 3 finite-state encoder for (0, 1)-RLL



- Labels represent input / output words.
- Input labels on edges with the same initial state are distinct.
- Output labelings of paths satisfy the $(0, 1)$-RLL constraint.
- There is a state-dependent decoder requiring look-ahead at most one codeword (encoder has finite anticipation $a = 1$).

Finite-State Coding Theorem

Theorem (Adler-Coppersmith-Hassner, 1983)

Let S be a constrained system with capacity $\text{cap}(S)$.
If $p/q \leq \text{cap}(S)$, *then there exists a rate* $p:q$ *finite-state encoder for* S *with finite anticipation.*

- Key implications:
  - Finite anticipation ensures a state-dependent decoder with finite look-ahead.
  - If $\text{cap}(S)$ is rational, then a capacity-achieving code exists.
  - If p and q are any integers with $p/q \leq \text{cap}(S)$, then an encoder using these block lengths exists.
- The proof is constructive: the state-splitting (ACH) algorithm.

E. The State-Splitting Algorithm

We now summarize the steps in the encoder construction procedure.

1) Find a deterministic FSTD G (or, more generally, an FSTD with finite local anticipation) which represents the given constrained system S (most constrained systems have a natural deterministic representation that is used to describe them in the first place).

2) Find the adjacency matrix $A = A(G)$ of G.

3) Compute the capacity $Cap(S)$ as $\log_2$ of the largest eigenvalue $\lambda(A)$ of A.

4) Select a desired code rate $p:q$ satisfying

$$Cap(S) \geq \frac{p}{q}$$

(one usually wants to keep p, q relatively small for complexity reasons).

5) Construct G^q.

6) Using the approximate eigenvector algorithm, find an $(A^q, 2^p)$-approximate eigenvector $\boldsymbol{v}$.

7) Eliminate all states i with $v_i = 0$, and restrict to an irreducible sink component H if necessary.

8) Find a basic $\boldsymbol{v}$-consistent partition for some state in H.

9) Find the basic $\boldsymbol{v}$-consistent state splitting corresponding to this partition, creating FSTD H' and approximate eigenvector $\boldsymbol{v}'$.

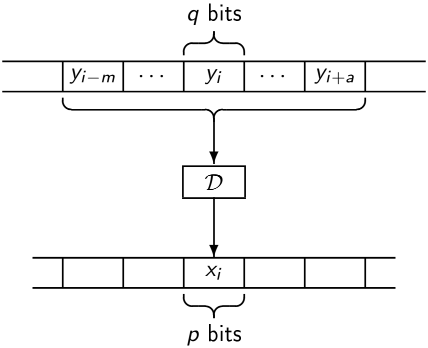
10) Iterate steps 8) and 9) until you obtain a graph H with minimum outdegree at least 2^p.

11) At each state of $\hat{H}$, delete all but 2^p outgoing edges and tag these edges with the binary p-blocks, one for each edge.

12) Congratulate yourself with a nice banana "split."

Sliding block decoder schematic

- State-dependent decoders can propagate errors.
- Sliding-block decoders limit error propagation.



- Decoder has look-ahead a and look-behind m.
- Error propagation is limited to $m + a + 1$ decoded words.

Decoder table for rate 2 : 3 (0, 1)-RLL encoder

current codeword y_i	next codeword y_{i+1}	decoded input $\mathcal{D}(y_i y_{i+1})$
010	—	11
011	101 or 111	01
011	010, 011, or 110	00
101	101 or 111	10
101	010, 011, or 110	00
110	—	10
111	101 or 111	11
111	010, 011, or 110	01

- Sliding-block decoder (shown only for valid codewords).
- Look-ahead $a = 1$ and look-behind $m = 0$.
- Error propagation limited to current and next input word.

Sliding-Block Code Theorem for Finite-Type Constraints

Theorem (Adler-Coppersmith-Hassner, 1983)

Let S be a finite-type constrained system with capacity $\text{cap}(S)$. If $p/q \leq \text{cap}(S)$, then there exists a rate $p : q$ finite-state encoder for S with a sliding-block decoder.

- A constrained system S is finite-type if it is defined by a finite list of forbidden words, e.g., (d, k)-RLL.
- The same construction works here too!
- Karabed-Marcus [1988] extended this to almost-finite-type constraints, including spectral-null constraints. The proof is harder and non-constructive.

State-splitting (ACH) algorithm

- Start with an irreducible, deterministic presentation G for constraint S, and $p/q \leq \mathrm{cap} S$.
- Apply Franaszek algorithm to find a nonnegative integer approximate eigenvector v satisfying

$$A_G^q \,\mathrm{v} \geq 2^{\mathrm{p}}\,\mathrm{v}.$$

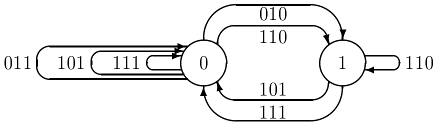
- Construct G^q, the qth power of G, representing S^q.
- Through a sequence of graph transformations (state splittings) guided by v, construct a graph H representing S^q that has at least 2^p outgoing edges at each state, i.e.

$$A_H 1 \geq 2^{\mathrm{p}} 1.$$

- Discard excess edges, merge states, if possible.
- Assign input words to edges, and start encoding!

Example: rate 2:3 (0, 1)-RLL

Graph G^3

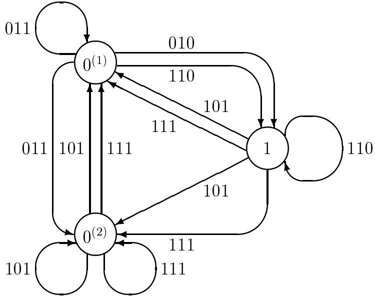


Approximate eigenvector

$$v^\top = [2 \quad 1]$$

$$A_G^3 v = \begin{bmatrix} 3 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \end{bmatrix} \geq 2^2 \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

Graph H after splitting state 0.

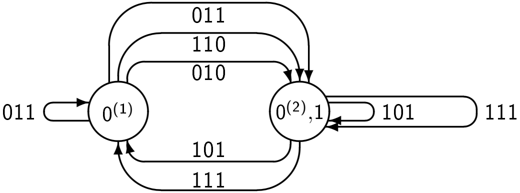


Approximate eigenvector

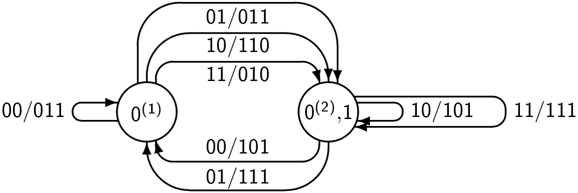
$$v^\top = [1 \quad 1 \quad 1]$$

$$A_H v = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 5 \end{bmatrix}.$$

- Excess edge deleted, states merged



- Input tags assigned



Comments on wordline page coding

- Wordline 3X3 cell patterns were eliminated by interleaved, rate 2:3 $(0,1)$-RLL coding on lower pages.

- With no extra coding on upper pages, the overall rate is $R = \frac{2}{3} + 1 \approx 1.6666$ bits/cell. (Highest possible rate is $R = C(0,1) + 1 \approx 1.6942$.)

- A constrained cell-level code over $\{0,1,2,3\}$ could eliminate 3-0-3, 3-1-3, 3-2-3 with highest possible rate $R \approx 1.9374$.

Joint wordline page coding

Cell levels	3-0-3	3-1-3	3-2-3
U	1 1 1	1 0 1	1 0 1
L	0 1 0	0 1 0	0 0 0

- The proposed scheme used coding on only the lower pages.
- Is there a more efficient scheme using jointly designed, but independent, codes on lower and upper pages?
- There is a formula for this joint capacity that allows us to answer that question. [Moision-Orlitsky-S, 2007]
- The answer is no!

Perron-Frobenius Theory

- The capacity formula, coding theorems, and code constructions make use of the Perron-Frobenius Theory of nonnegative matrices.
- The P-F theory also provides the mathematical justification of the power method used by Brin and Page to iteratively compute Google's PageRank ranking of Web pages!
- To state the results, we need two definitions:
  - A nonnegative matrix A is irreducible if for any row-column index (u, v), there is an integer $n_{u,v}$ such that $(A^{n_{u,v}})_{u,v} > 0$.
  - An irreducible matrix A is primitive if the integer $n_{u,v}$ above can be chosen independent of u, v.

Theorem (Perron-Frobenius)

An irreducible matrix A has an eigenvalue λ such that:

- λ *is real and positive*
- *associated with* λ *are strictly positive right and left eigenvectors,* x *and* $\mathrm{y}^\top$*, unique up to scaling*
- $|\lambda| \geq |\mu|$ *for any other eigenvalue of A, with strict inequality if A is primitive; i.e.,* λ *is the spectral radius* $\rho(A)$
- λ *is a simple root of the characteristic polynomial of A*
- *If A is primitive, and* $\mathrm{y}^\top\mathrm{x} = 1$*, then* $\lim_{k\to\infty}(\lambda^{-1}A)^k = \mathrm{xy}^\top$.

Example

- Let $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$; characteristic polynomial $x^2 - x - 1$.
- The eigenvalues are $\lambda = \lambda(A) = \frac{1+\sqrt{5}}{2}$ and $\mu = \frac{1-\sqrt{5}}{2}$.
- Right and left eigenvectors associated with λ are given by:

$$\mathrm{x}^\top = [\lambda \quad 1] \qquad \mathrm{y} = [\lambda \quad 1].$$

- The characteristic polynomial factors as

$$x^2 - x - 1 = (x - \lambda)(x - \mu).$$

- The normalized product converges to

$$\lim_{k\to\infty} (\lambda^{-1} A)^k = \frac{1}{1+\lambda^2} \begin{bmatrix} \lambda^2 & \lambda \\ \lambda & 1 \end{bmatrix}$$

Google's PageRank

- Let $\{P_i\}$ be the set of all web pages, $i = 1, \ldots, 4.77 \times 10^9$, each with PageRank $\pi(i)$, normalized such that $\sum_i \pi(i) = 1$.
- The PageRank vector $\pi = (\pi(\mathrm{i}))$ satisfies

$$\pi^\top = \pi^\top \cdot \mathrm{M}$$

where M is a primitive, stochastic matrix that reflects the link structure among all pages, as well as some aspects of typical web surfing behavior.

- The equation is solved using an iterative procedure

$$\pi^{(\mathrm{k}+1)\top} = \pi^{(\mathrm{k})\top} \cdot \mathrm{M}$$

with $\pi^{(0)\top} = [1/\mathrm{n}, \ldots, 1/\mathrm{n}]$.

- Convergence follows from the P-F Theorem!

2D: Row-by-row bitline coding

Bitline constraints

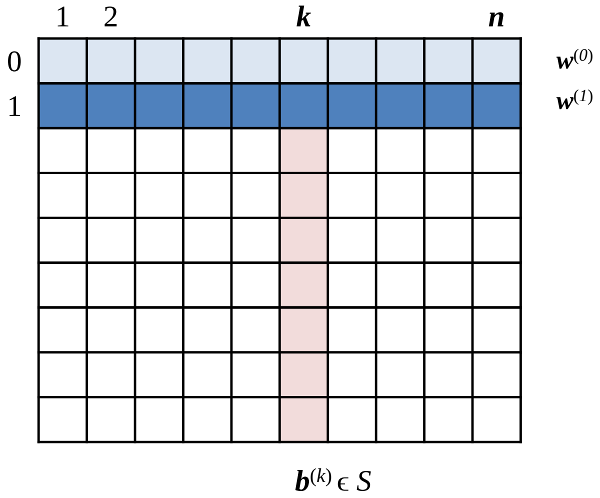
- Bitline ICI causes more errors than wordline ICI.
- A code enforcing $(0,1)$-RLL constraints on interleaved bitline lower bits eliminates bitline 3X3 cell patterns.
- We will construct such a code compatible with row-by-row programming.
- The construction can achieve a rate close to $C(0,1)$.

Row-by-cow coding for bitline $(0, 1)$-RLL

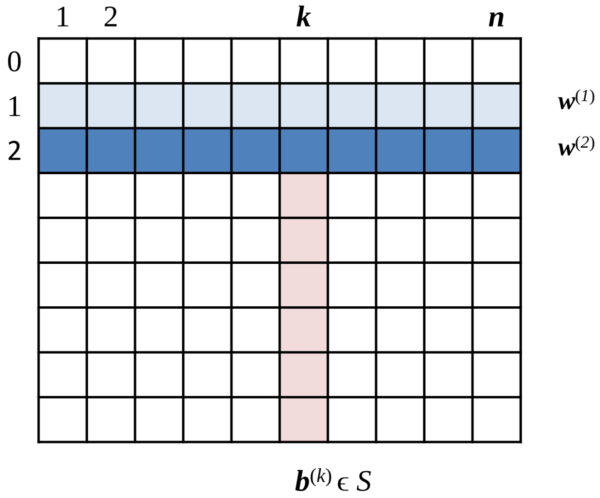
- The row-by-row code construction consists of 2 steps:
  - Step 1: Probabilistic analysis
  - Step 2: Code design using constant-weight codes
- The encoder has the following properties:
  - Encoding is row-by-row and fixed rate.
  - Encoding / decoding a row requires the previous row.
  - The code rate can approach the capacity $C(0, 1)$ (as the number of bitlines approaches infinity).

[Buzaglo-Yaakobi-S, 2015]

Row-by-row encoder schematic



Row-by-row encoder schematic



Constant weight codes

- $\mathbb{C}(m, w)$ denotes the constant weight code that consists of all binary sequences of length m and weight w.
- For example, the codebook for $\mathbb{C}(3, 2)$ is:

0	1	1
1	0	1
1	1	0

- The asymptotic encoding rate of the code $\mathbb{C}(n, \lfloor \beta n \rfloor)$ is

$$C(\beta) = \lim_{n\to\infty} (1/n) \log_2 |\mathbb{C}(n, \lfloor \beta n \rfloor)| = h_2(\beta);$$

e.g., $C(\frac{2}{3}) = h_2(\frac{2}{3}) = -\frac{2}{3}\log_2(\frac{2}{3}) - \frac{1}{3}\log_2(\frac{1}{3}) \approx 0.9183.$

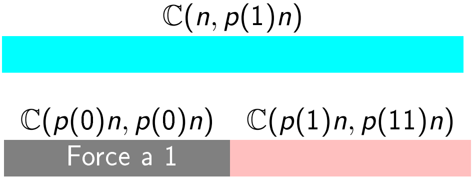
- We use 2 length-n codes built from various $\mathbb{C}(m, w)$:

$$\begin{aligned}\mathbb{C}^1 &= \mathbb{C}(n, p(1)n) \\ \mathbb{C}^2 &= \mathbb{C}(p(0)n, p(0)n) \times \mathbb{C}(p(1)n, p(11)n)\end{aligned}$$

- $\mathbb{C}(p(0)n, p(0)n)$ contains only the all-ones codeword $[1 \ldots 1]$.
- Set $p(0) = 1/4,\ p(1) = 3/4,\ p(11) = 1/2,\ n$ a multiple of 4.
- Asymptotic rate for $\mathbb{C}^1$ and $\mathbb{C}^2$

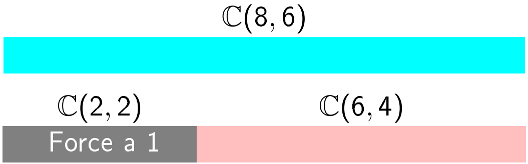
$$\begin{aligned}R(\mathbb{C}^1) &= C\left(p(1)\right) = h_2\left(3/4\right) \approx 0.8112. \\ R(\mathbb{C}^2) &= p(1)C\left(p(11)/p(1)\right) = \frac{3}{4}h_2\left(2/3\right) \approx 0.6887.\end{aligned}$$

Encoding



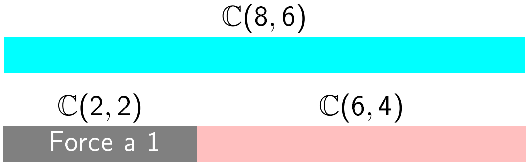
- WL_1: Encode using $\mathbb{C}(n, p(1)n)$.
- WL_i, $i \geq 2$:
  Find index sets I_0, I_1 where values in WL_{i-1} are 0, 1.
  For corresponding index sets in WL_i, encode using $\mathbb{C}(p(0)n, p(0)n)$ and $\mathbb{C}(p(1)n, p(11)n)$

Encoding example: $n = 8$



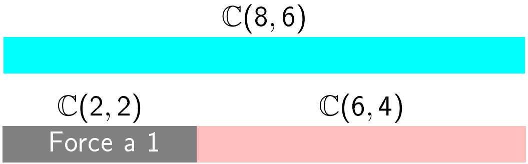
WL_1	1	1	0	1	0	1	1	1

Encoding example: $n = 8$



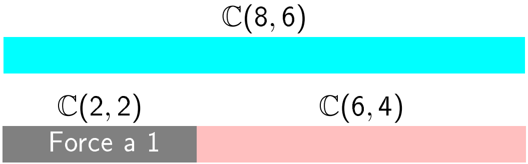
WL_1	1	1	0	1	0	1	1	1
WL_2	1	0	1	1	1	1	0	1

Encoding example: $n = 8$



WL_1	1	1	0	1	0	1	1	1
WL_2	1	0	1	1	1	1	0	1
WL_3	1	1	1	0	1	0	1	1

Encoding example: $n = 8$



WL_1	1	1	0	1	0	1	1	1
WL_2	1	0	1	1	1	1	0	1
WL_3	1	1	1	0	1	0	1	1
WL_4	1	0	1	1	0	1	1	1

- Each row has the same distribution of 0s and 1s.
- The bitline $(0, 1)$-RLL constraint is enforced.

Decoding

WL_1	1	1	0	1	0	1	1	1
WL_2	1	0	1	1	1	1	0	1
WL_3	1	1	1	0	1	0	1	1
WL_4	1	0	1	1	0	1	1	1

- WL_1: Decode using $\mathbb{C}(n, p(1)n)$.
- WL_i, $i \geq 2$:

  Find index set I_1 where value in WL_{i-1} is 1.
  For corresponding index set in WL_i, decode using $\mathbb{C}(p(1)n, p(11)n)$.

Stationary Markov chains

- The probabilities used in the construction come from a stationary Markov chain on the $(0,1)$-RLL constraint graph.
- Stationary Markov chain $\mathcal{P} = (Q, \pi)$ on graph G:
  - transition matrix $Q = (Q_{u,v})_{u,v \in V}$
  - stationary probability vector $\pi = (\pi_{\mathrm{u}})_{\mathrm{u} \in \mathrm{V}}$
- Stationarity condition

$$\pi^\top \cdot \mathrm{Q} = \pi^\top .$$

- The entropy of $\mathcal{P}$ is given by

$$H(\mathcal{P}) = -\sum_{u \in V} \pi_u \sum_{u \to v} Q_{u,v} \log_2 Q_{u,v} .$$

Probabilistic characterization of capacity

Theorem (Shannon, 1948)

Let S be a constraint with irreducible, lossless presentation G. Then

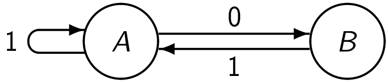
$$\text{cap}(S) = \sup_{\mathcal{P}} H(\mathcal{P})$$

where the sup is taken over all stationary Markov chains $\mathcal{P}$ on G.

- Let x be a right eigenvector of A_G associated with $\lambda = \lambda(A_G)$.
- The unique maxentropic Markov chain $\mathcal{P}^* = (Q^*, \pi^*)$ has

$$Q^*_{u,v} = \frac{x_v}{\lambda x_u}$$

as transition probability for edge $u \to v$.



- Right eigenvector $x = [\lambda \;\; 1]$.

- Transition probabilities

$$Q^* = \begin{bmatrix} \lambda^{-1} & \lambda^{-2} \\ 1 & 0 \end{bmatrix} \approx \begin{bmatrix} 0.618 & 0.382 \\ 1 & 0 \end{bmatrix}$$

- Stationary state probabilities

$$\pi^* = \left[\frac{\lambda^2}{1+\lambda^2} \;\; \frac{1}{1+\lambda^2} \right] \approx [0.724 \quad 0.276]$$

Probabilistic analysis

- Approximate $\mathcal{P}^*$ by stationary Markov chain $\mathcal{P} = (Q, \pi)$:
  - $\pi_u n$ is an integer, for all $u \in V$
  - $(\pi_u Q_{u,v})n$ is an integer, for all $u, v \in V$.
- Define $p(x) \stackrel{\text{def}}{=} \Pr(x)$ and $p(11) \stackrel{\text{def}}{=} \Pr(x\ 1)$, for $x \in \{0, 1\}$:
- Then $p(x)n$ and $p(x1)n$ are also integers.
- For large enough n, we can find $\mathcal{P}$ with $H(\mathcal{P}) \approx C(0, 1)$.

- Conditional edge probability matrix $Q = (Q_{u,v})$

$$Q = \begin{bmatrix} 2/3 & 1/3 \\ 1 & 0 \end{bmatrix} \approx \begin{bmatrix} 0.618 & 0.382 \\ 1 & 0 \end{bmatrix}$$

- Stationary state probability vector π

$$\pi = [3/4 \quad 1/4] \approx [0.724 \quad 0.276]$$

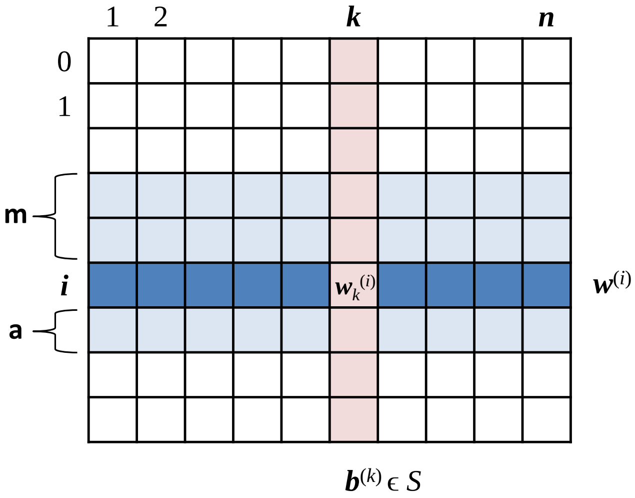
- $p(0) = 1/4,\ p(1) = 3/4 \quad ; \quad p(01) = 1/4,\ p(11) = 1/2.$

- $H(\mathcal{P}) = \frac{3}{4} h_2(\frac{2}{3}) \approx 0.6887.$

Generalization: n-track parallel encoder

- Let S be a constrained system.
- We define a rate-R, n-track parallel encoder for S as follows:
  - For row $i = 0, 1, 2, \ldots$, the encoder input x is $n \cdot R$ bits.
  - For row $i = 0, 1, 2, \ldots$, the encoder output is a codeword $\mathrm{w}^{(i)}$ of length n. (Encoding may depend on a finite number of previously written codewords.)
  - For column $k = 1, 2, \ldots, n$, the column word $\mathrm{b}^{(k)}$ is in S.
- The encoder is (m, a) sliding-block-decodable if, for some $m, a \geq 0$, we can decode row codeword $\mathrm{w}^{(i)}$, $\forall\, i \geq m$, from row codewords $\mathrm{w}^{(i-m)}, \ldots, \mathrm{w}^{(i)}, \ldots \mathrm{w}^{(i+a)}$.

Sliding-block decodable n-track parallel encoder



Parallel encoder for bitline constraint S

Theorem (Tal-Etzion-Roth, 2009)

Let G be a deterministic graph with memory m representing S.

For sufficiently large n, one can construct an $(m, 0)$ sliding-block decodable n-track parallel encoder for S at rate R, where

$$R \geq \mathrm{cap}(S)\left(1 - \frac{c}{n}\right) - O\left(\frac{\log n}{n}\right)$$

where c is a constant that depends on the graph G.

Moreover, the encoder requires knowledge of no more than the preceding m codewords.

Remarks

- There is a general method for finding an approximating Markov process satisfying the integrality conditions.
- There are efficient encoding and decoding algorithms for constant weight codes.
- This technique can be used in conjunction with a wordline ICI-mitigating constrained code on wordline upper pages.
- Combining the $(0,1)$-RLL row-by-row code on bitline lower bits with a (conventional) $(1,\infty)$-RLL code on wordline upper pages eliminates all bitline and wordline 3X3 cell patterns.
- Asymptotic rate $R \approx 0.6942 + 0.6942 = 1.3884$ bits/cell.

Capacity of discrete channel with noise

$$C = \mathrm{Max}(H(x) - H_y(x))$$

For discrete noiseless channel, $H_y(x) = 0$, so

$$C = \mathrm{Max}\ H(x)$$

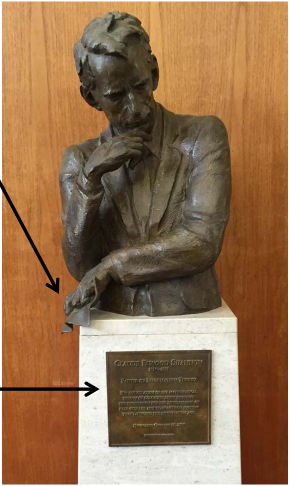
CLAUDE ELWOOD SHANNON
1916 – 2001

FATHER OF INFORMATION THEORY

HIS FORMULATION OF THE MATHEMATICAL THEORY OF COMMUNICATION PROVIDED THE FOUNDATION FOR THE DEVELOPMENT OF DATA STORAGE AND TRANSMISSION SYSTEMS THAT LAUNCHED THE INFORMATION AGE.

DEDICATED OCTOBER 16, 2001

EUGENE DAUB, SCULPTOR



2D: Combined Wordline and Bitline Coding

Combined wordline and bitline coding

- Enforcing the “no 1X1” constraint on upper bits along both wordlines and bitlines will eliminate 3X3 patterns in both directions.

- This translates to enforcing $(1, \infty)$-RLL constraints on rows and columns of 4 interleaved subarrays.

$*$	$\square$	$*$	$\square$
$\times$	$\triangle$	$\times$	$\triangle$
$*$	$\square$	$*$	$\square$
$\times$	$\triangle$	$\times$	$\triangle$

- Each interleaved subarray satisfies 2D $(1, \infty)$-RLL constraints.

2D (d, k)-RLL constraints

- The 2D (d, k)-RLL constrained system is the set of $m \times n$ arrays with each row and each column satisfying the (d, k)-RLL constraint.

- Example: 2D $(1, \infty)$-RLL (hard-square model)

0	1	0	0	1	0
1	0	1	0	0	1
0	0	0	1	0	0
1	0	0	0	1	0
0	0	0	1	0	0

- We can define other 2D constrained systems using other "local" constraints.

Capacity of 2D constraints

- The capacity of a 2D constrained system S is the growth rate of the number of $m \times n$ arrays, $N(m, n; S)$:

$$\mathrm{cap}_2(S) = \limsup_{m,n \to \infty} \frac{\log_2 N(m, n; S)}{mn}$$

- As for 1D constraints, the limit exists.
- The exact capacity is known for very few 2D constraints, e.g.,
  - hard-hexagon model [Baxter, 1980]
  - path-cover constraint [Schwartz-Bruck, 2008]

- Let $\text{cap}_2(d, k)$ denote capacity for 2D (d, k)-RLL.

Capacity of 2D RLL constraints

- Let $\text{cap}_2(d, k)$ denote capacity for 2D (d, k)-RLL.
- Clearly, $\text{cap}_2(0, \infty) = 1$ and $\text{cap}_2(0, 1) = \text{cap}_2(1, \infty)$.

Capacity of 2D RLL constraints

- Let $\text{cap}_2(d,k)$ denote capacity for 2D (d,k)-RLL.
- Clearly, $\text{cap}_2(0,\infty) = 1$ and $\text{cap}_2(0,1) = \text{cap}_2(1,\infty)$.
- There is no known general formula for computing $\text{cap}_2(d,k)$.

Capacity of 2D RLL constraints

- Let $\text{cap}_2(d,k)$ denote capacity for 2D (d,k)-RLL.
- Clearly, $\text{cap}_2(0,\infty) = 1$ and $\text{cap}_2(0,1) = \text{cap}_2(1,\infty)$.
- There is no known general formula for computing $\text{cap}_2(d,k)$.
- But, the zero-capacity region of 2D RLL constraints is known!

Theorem (Kato-Zeger, 1999)

For every $d \geq 1$ and every $k > d$,

$$\text{cap}_2(d, k) = 0 \iff k = d + 1.$$

Theorem (Kato-Zeger, 1999)

For every $d \geq 1$ and every $k > d$,

$$\text{cap}_2(d, k) = 0 \iff k = d + 1.$$

- Examples:
  - $\text{cap}_2(1, 2) = 0$.
  - $\text{cap}_2(2, 4) > 0$.

Zero-capacity region for 2D (d, k)-RLL

Theorem (Kato-Zeger, 1999)

For every $d \geq 1$ and every $k > d$,

$$\mathrm{cap}_2(d, k) = 0 \iff k = d + 1.$$

- Examples:
  - $\mathrm{cap}_2(1, 2) = 0$.
  - $\mathrm{cap}_2(2, 4) > 0$.
- This is strange, because $C(1, 2) = C(2, 4) \approx 0.4507$.

$\text{cap}_2(1,2) = 0$

- Let $X = [x_{i,j}]$, $(i,j) \in \mathbb{Z}^2$ be an infinite 2D $(1,2)$-RLL array.

$\mathrm{cap}_2(1,2) = 0$

- Let $X = [x_{i,j}]$, $(i,j) \in \mathbb{Z}^2$ be an infinite 2D $(1,2)$-RLL array.
- Any pattern 1 0 0 1 in a row has 2 possible configurations

			1	0			
			0	1			
1	0	1	0	0	1	0	1
			1	0			
			0	1			

			0	1			
			1	0			
1	0	1	0	0	1	0	1
			0	1			
			1	0			

$\text{cap}_2(1,2) = 0$

- Let $X = [x_{i,j}]$, $(i,j) \in \mathbb{Z}^2$ be an infinite 2D (1, 2)-RLL array.
- Any pattern 1 0 0 1 in a row has 2 possible configurations

			1	0			
			0	1			
1	0	1	0	0	1	0	1
			1	0			
			0	1			

			0	1			
			1	0			
1	0	1	0	0	1	0	1
			0	1			
			1	0			

- The row determines the rest of the array by diagonal or anti-diagonal extension:

$$x_{i,j} = x_{0,i+j},\ \forall i,j \qquad \text{or} \qquad x_{i,j} = x_{0,i-j},\ \forall i,j$$

- The number of $m \times n$ constrained arrays grows exponentially in n, but not mn.

$\text{cap}_2(1,2) = 0$

- Let $X = [x_{i,j}]$, $(i,j) \in \mathbb{Z}^2$ be an infinite 2D $(1,2)$-RLL array.
- Any pattern 1 0 0 1 in a row sits in 2 possible configurations

1	0	0	1	0	1	0	
0	1	0	0	1	0	1	0
1	0	1	0	0	1	0	1
0	1	0	1	0	0	1	0
	0	1	0	1	0	0	1

	0	1	0	1	0	0	1
0	1	0	1	0	0	1	0
1	0	1	0	0	1	0	1
0	1	0	0	1	0	1	0
1	0	0	1	0	1	0	

- The row determines the rest of the array by diagonal or anti-diagonal extension:

$$x_{i,j} = x_{0,i+j}, \ \forall i,j \qquad \text{or} \qquad x_{i,j} = x_{0,i-j}, \ \forall i,j$$

- The number of $m \times n$ constrained arrays grows exponentially in n, but not mn.

- Matrix methods, exploiting symmetry properties of the constraint, yield very good bounds on $\mathrm{cap}_2(1, \infty)$:

$$0.587891161775 \leq \mathrm{cap}_2(1, \infty) \leq 0.587891161868.$$

[Calkin-Wilf, 1998],[Forchhammer-Justesen, 1999], [Nagy-Zeger, 2000]

- A 2D $(1, \infty)$-RLL code on upper bits, with uncoded lower bits, could have overall rate:

$$R \approx 0.587891161 + 1 \approx 1.5878 > 1.3884$$

beating the row-by-row method.

Strip encoder

- View 2D constrained array as stack of height-h strips.
- Encode data into 1D sequences of column symbols in Σ^h using 1D encoder (designed, e.g., by state-splitting).
- Glue strips together with fixed-height merging strips.

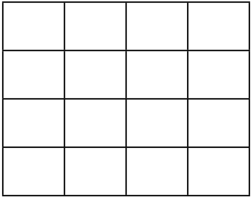
↑	1	0	1	0	1	0
h	0	0	0	1	0	1
↓	1	0	1	0	1	0
	0	0	0	0	0	0
	0	0	1	0	1	0
	0	0	0	1	0	1
	1	0	1	0	0	0

[Etzion, 1997]

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0



- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0			

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		
0			

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		0
0			

- Efficiency $R/\mathrm{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		0
0		1	

- Efficiency $R/\mathrm{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		0
0		1	
	0		

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Rate-1/2 encoder for 2D $(1, \infty)$

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0		1	
	1		0
0		1	
	0		0

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

- Fixed rate $R = 1/2$ encoder.
- Write data raster fashion along odd diagonals.
- Insert 0s elsewhere.

Data: 0 1 1 0 0 1 0 0

0	0	1	0
0	1	0	0
0	0	1	0
0	0	0	0

- Efficiency $R/\text{cap}_2(1, \infty) \approx 0.85$.

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, “stuff” a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, “stuff” a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1			

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1			
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0		
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, “stuff” a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0		
1	0			

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0		
1	0			
0				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	
1	0			
0				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	
1	0	1		
0				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	
1	0	1		
0	1			

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	
1	0	1		
0	1			
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	0
1	0	1		
0	1			
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	0
1	0	1	0	
0	1			
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	0
1	0	1	0	
0	1	0		
1				

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1, \infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	0
1	0	1	0	
0	1	0		
1	0			

[S-Wolf, 1998]

Bit-stuffing encoder for 2D $(1,\infty)$

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, "stuff" a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

0	1	0	0	0
1	0	1	0	1
0	1	0	0	0
1	0	1	0	0

[S-Wolf, 1998]

- Write into array raster fashion along successive diagonals.
- If the written bit above or to the left is 1, “stuff” a 0.
- If not, write the next data bit.
- Decoder proceeds in same order, discarding stuffed bits.
- Example: 0 1 1 0 1 1 1 0 1 0 1 0

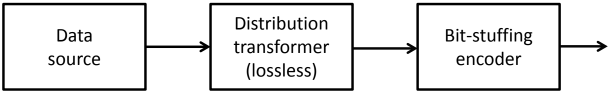
0	1	0	0	0
1	0	1	0	1
0	1	0	0	0
1	0	1	0	0

[S-Wolf, 1998]

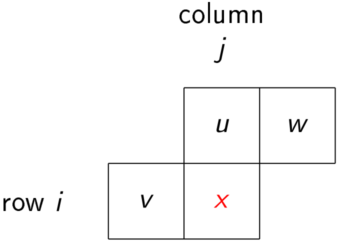
Remarks about bit-stuffing encoders

- Bit-stuffing is variable-rate.
- Biasing input can increase the code rate (use fewer 1s).
- Bit-stuffing can be applied to other 2D constraints.
- Encoder rate provides a lower bound on capacity.
- Lower bounds on encoder rate can be effectively computed. [Tal-Roth, 2010]

Biased bit-stuffing encoder block diagram



- Lossless distribution transformer $\mathcal{E}$ converts i.i.d. equiprobable bits to i.i.d. biased bits with $Pr(0) = q$.
- Rate penalty $h_2(q)$.
- Bit-stuffing encoder accepts transformer output and writes to array using bit-stuffing rules.



- Encoding rule for position x:

$$x = \begin{cases} 0 & \text{if } u = 1 \text{ or } v = 1 \\ \text{next bit from } \mathcal{E} & \text{otherwise.} \end{cases}$$

- The rate $\mathcal{R}(q)$ can be determined exactly.

Exact analysis of 2D $(1, \infty)$ encoder

- Let $\gamma = Pr(u = v = 0)$, the probability that x is not stuffed.
- Average rate: $\mathcal{R}(q) = h_2(q)\gamma$ where

$$\gamma = \frac{(4-3q) - \sqrt{(4-3q)^2 - 4(1-q)(4-3q)}}{2(1-q)(4-3q)}.$$

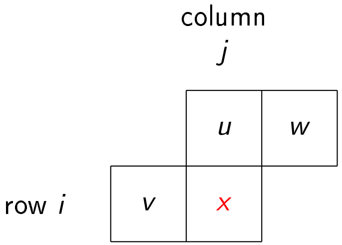
- Find $q^{opt} = 1 - p^{opt}$ to maximize rate $\mathcal{R}(q)$:

$$q^{opt} \approx 0.6444 \Longrightarrow \mathcal{R}(q^{opt}) = 0.5830\ldots$$

- Efficiency:

$$\frac{\mathcal{R}(q^{opt})}{\mathrm{cap}_2(1,\infty)} \geq 0.9917.$$

Enhanced 2D $(1, \infty)$ bit-stuffing encoder



- Distribution transformers, $\mathcal{E}_0$ and $\mathcal{E}_1$, biases q_0 and q_1.
- Encoding rule for position x:

$$x = \begin{cases} 0 & \text{if } u = 1 \text{ or } v = 1 \\ \text{next bit from } \mathcal{E}_w & \text{otherwise.} \end{cases}$$

- Rate $\mathcal{R}(q_0, q_1)$ can again be determined exactly.

Efficiency of enhanced 2D $(1,\infty)$ encoder

- Optimize parameters q_0 and q_1:

$$q_0^{opt} \approx 0.6718, \quad q_1^{opt} \approx 0.6669$$

$$\Longrightarrow \mathcal{R}(q_0^{opt}, q_1^{opt}) \approx 0.587277.$$

- Efficiency of enhanced 2D $(1,\infty)$ bit-stuffing encoder:

$$\frac{\mathcal{R}(q_0^{opt}, q_1^{opt})}{C_2(1,\infty)} \geq 0.9989$$

- Conditioning on more of the past much harder to analyze.

[Roth-S-Wolf, 2001]

Further remarks

- Bit-stuffing encoders have been studied for other 2D constraints: (d, ∞), "no-isolated-bit", "checkerboard".
- A general method based upon linear programming for bounding the rate of 2D bit-stuffing encoders has provided improved lower bounds on capacity of some 2D constraints.
- Further results on capacity bounds, positive capacity regions, and asymptotic capacity for multidimensional constraints have been obtained.
- Much remains to be done in the area of multidimensional constrained coding.

- Did Shannon say anything about 2D constrained systems?

> *"The redundancy of a language is related to the existence of crossword puzzles. If the redundancy is zero any sequence of letters is a reasonable text in the language and any two-dimensional array of letters forms a crossword puzzle. If the redundancy is too high the language imposes too many constraints for large crossword puzzles to be possible."*

Shannon's crossword puzzles

- More specifically ...

> *"A more detailed analysis shows that if we assume the constraints imposed by the language are of a rather chaotic and random nature, large crossword puzzles are just possible when the redundancy is 50%. If the redundancy is 33%, three dimensional crossword puzzles should be possible, etc."*

- Translation of terms:

  Language $\Rightarrow$ constrained system S.

  Redundancy $\Rightarrow$ $r(S) = 1 - \text{cap}(S)$.

  Crossword puzzles $\Rightarrow$ 2D constraint $S^{\otimes 2}$ consisting of $m \times n$ arrays with rows and columns in S.

  Large crossword puzzles possible $\Rightarrow$ number of $m \times n$ arrays grows exponentially in mn, $\text{cap}_2(S^{\otimes 2}) > 0$.

- If we add

  Chaotic and random $\Rightarrow$ rows and columns of arrays in $S^{\otimes 2}$ are statistically independent.

then Shannon's statement can be rederived.

(d, k) crossword puzzles

- Application to (d, k)-RLL constraints:
  - $\text{cap}(1,2) = \text{cap}(2,4) \approx 0.4057$.
  - $r(1,2) = r(2,4) \approx 0.5943 > 50\% \Rightarrow$ no large puzzles
  - $\text{cap}_2(1,2) = 0$ but $\text{cap}_2(2,4) > 0$.
- The analysis does not seem to apply to (d, k) constraints.
- Conclusion:

(d, k) crossword puzzles deserve further investigation!

Concluding Remarks

Concluding remarks

- Constrained coding is interesting, practical, and fun.
- New generations of storage technologies will need them.
- There are many other research directions beyond those discussed here:
  - Constrained error-correcting codes
  - Constrained codes with unconstrained positions
  - Constrained codes with global constraints
  - Endurance codes, shaping codes, semiconstrained systems.

-.... .- -. -.-   -.-- --- ..-

t h a n k   y o u

Introduction

- C.E. Shannon, "The mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423, 623–656, July, October 1948.

Flash Memory Basics

- Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582-1595, April 2010.
- E. Yaakobi, J. Ma, L. Grupp, P. H. Siegel, S. Swanson, and J. K. Wolf, "Error characterization and coding schemes for flash memories," in *Proc. IEEE Globecom Workshops*, Dec. 2010, pp. 1856-1860.
- Y. Cai, E.F. Haratsch, O. Mutlu, and K. Mai, "Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis," in *Proc. DATE*, March 2012, pp. 521-526.

Flash Memory Basics (cont.)

- J. Moon, J. Lo, S. Lee, S. Kim, S. Choi, and Y. Song,"Statistical characterization of noise and interference in NAND flash memory," in *IEEE Trans. Circuits and Systems - I: Regular Papers*, vol. 60, no. 8, August 2013, pp. 2153-2164.
- Y. Cai, E.F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modeling," in *Proc. DATE*, March 2013, pp.1285-1290.
- Y. Cai, O. Mutlu, E.F. Haratsch, and K. Mai, "Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation," in *Proc. IEEE ICCD*, June 2013, pp. 123-130.
- J. Cooke, "The inconvenient truths about NAND flash memory." *Micron MEMCON 7*, 2007.

Wordline Page Coding

Texts on constrained coding

- B.H. Marcus, R.M. Roth, P.H. Siegel, *An Introduction to Coding for Constrained Systems*, Draft text (Fifth Edition), October 2001. (available online)
- K.A.S. Immink, *Codes for Mass Data Storage Systems*, Shannon Foundation Publishers. The Netherlands, 1999, 2004.

Expository articles and surveys on constrained coding

- B.H. Marcus, P.H. Siegel, and J.K. Wolf, "Finite-state modulation codes for data storage," *IEEE J. Sel. Areas Comm.*, vol. 10, no. 1, January 1992, pp. 5–37.

Wordline Page Coding (cont.)

- K.A.S. Immink, P.H. Siegel, and J.K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory, Special Commemorative Issue,* vol. 44, no. 6, pp. 2260-2299, October 1998.
- B.H. Marcus, R.M. Roth, P.H. Siegel, "Constrained systems and coding for recording channels," *Handbook of Coding Theory, V.S. Pless and W.C. Huffman (Editors),* Elsevier, Amsterdam (1998), pp. 1635–1764.

The ACH paper

- R.L. Adler, D. Coppersmith, M. Hassner, "Algorithms for sliding block codes — an application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory,* vol. 29, no. 1, January 1983, pp. 5–22.

Wordline Page Coding (cont.)

Articles on constrained codes for flash memory

- A. Berman and Y. Birk, "Constrained flash memory programming," *Proc. IEEE Symp. Inf. Theory,* July–August, 2011, pp. 2128–2132.
- R. Motwani, "Hierarchical constrained coding for floating-gate to floating-gate coupling mitigation in flash memory," in *Proc. IEEE Globecom*, December 2011.
- A. Berman and Y. Birk, "Low-complexity two-dimensional data encoding for memory inter-cell interference reduction," *Proc. 27th Conv. IEEE Israel (IEEEI)*,November 2012.

Wordline Page Coding (cont.)

- Y. Kim, B. Kumar, K. L. Cho, H. Son, J. Kim, J. J. Kong, and J. Lee,"Modulation coding for flash memories," in *Proc. ICNC*, Jan. 2013, pp. 961-967.
- M. Qin, E. Yaakobi, and P. H. Siegel,"Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories," *IEEE JSAC*, vol. 32, no. 5, pp. 836-846, May 2014.
- V. Taranalli, H. Uchikawa, and P.H. Siegel, "Error analysis and inter-cell interference mitigation in multi-level cell flash memories," in *Proc. IEEE ICC*, June 2015.

Joint Wordline Page Coding

Distance-enhancing codes for PRML

- R. Karabed and P. Siegel, "Matched Spectral Null Codes for Partial Response Channels," *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pt. II, pp. 818-855, May 1991.
- R. Karabed and P. H. Siegel, "Coding for higher order partial response channels," in *Proc. SPIE* (M. R. Raghuveer, S. A. Dianat, S. W. McLaughlin, and M. Hassner, eds.), vol. 2605, (Philadelphia, PA, USA), pp. 92–102, Oct. 1995.
- J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. Magn.*, vol. 32, pp. 3992–3994, Sept. 1996.
- W. G. Bliss, "An 8/9 rate time-varying trellis code for high density magnetic recording," *IEEE Trans. Magn.*, vol. 33, pp. 2746–2748, Sept. 1997.

Joint Wordline Page Coding (cont.)

- B.E. Moision, P.H. Siegel, and E. Soljanin, " Distance-Enhancing Codes for Digital Recording," *IEEE Trans. Magn.*, vol. 34, no. 1, pp. 69-74, Jan. 1998.
- S.A. Atlekar, M. Berggren, B.E. Moision, P.H. Siegel, and J.K. Wolf, "Error Event Characterization on Partial Response Channels," *IEEE Trans. Inform. Theory,* vol. 45, no. 1, pp. 241-247, January 1999.
- R. Karabed, P.H. Siegel, and E. Soljanin, "Constrained Coding for Binary Channels with High Intersymbol Interference," *IEEE Trans. Inform. Theory,* vol. 45, pp. 1777-1797, September 1999.
- T. Lei Poo and B. H. Marcus, "Time-varying maximum transition run constraints," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4464–4480, Oct. 2006.

Joint Wordline Page Coding (cont.)

Codes avoiding specified differences

- B.E. Moision, A. Orlitsky, and P.H. Siegel, "On Codes That Avoid Specified Differences," *IEEE Trans. Inform. Theory*, vol. 47, no. 1, pp. 433–442, January 2001.
- V.D. Blondel, R. Jungers, and V. Protasov, "On the complexity of computing the capacity of codes that avoid forbidden difference patterns," *IEEE Trans. Inform. Theory*, vol. 52, no. 11, pp. 5122–5127, November 2006.
- B.E. Moision, A. Orlitsky, and P.H. Siegel, " On Codes with Local Joint Constraints," *Lin. Alg. Appl.*, vol. 422, iss. 2-3, pp. 442–454, April 15, 2007.

Joint Wordline Page Coding (cont.)

Joint spectral radius

- G. C. Rota and G. Strang, "A note on the joint spectral radius," *Indagationes Mathematicae,* vol. 22, pp. 379–381, 1960.
- I. Daubechies and J. C. Lagarias, "Two-scale difference equations II. Local regularity, infinite products of matrices and fractals," *SIAM J. Math. Anal.,* vol. 23, pp. 1031–1079, 1992.
- M. A. Berger and Y. Wang, "Bounded semi-groups of matrices," *Linear Algebra Applications,* vol. 166, pp. 21–27, 1992.
- G. Gripenberg, "Computing the joint spectral radius," *Linear Algebra Applications,* vol. 234, pp. 43–60, 1996.

Row-by-Row Bitline Coding

- S. Halevy and R. M. Roth, "Parallel constrained coding with application to two-dimensional constraints," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1009–1020, 2002.
- I. Tal, T. Etzion, and R. Roth, "On row-by-row coding for 2-D constraints," *IEEE Trans. Inform. Theory*, vol. 55, pp. 3565–3576, 2009.
- S. Buzaglo, P. H. Siegel, and E. Yaakobi, "Coding schemes for inter-cell interference in flash memory," in *Proc. IEEE Int. Symp. Inform. Theory,* pp. 3564–3570, Hong Kong, June 14–19, 2015

Combined Wordline and Bitline Coding

Capacity of 2D constraints

- R.M. Robinson, "Undecidability and nonperiodicity for tilings of the plane," *Inventiones Math.*, 12 (1971), 177–209.
- J. Ashley and B. Marcus, "Constant-weight/lowpass modulation codes for holographic recording," Research Report RJ-10089 (91905), IBM Research Laboratory, San Jose, California, 1997.
- N. Calkin and H.S. Wilf, "The number of independent sets in a grid graph," *SIAM J. Discrete Math.*, 11 (1997), 54–60.
- T. Etzion, "Cascading Methods for Runlength-Limited Arrays," *IEEE Trans. Inform. Theory*, IT-43 (1997), 319-324.
- J. Ashley and B. Marcus, "Two-dimensional low-pass filtering codes," *IEEE Trans. Commun.*, vol. 46, no. 6, pp. 724–727, June 1998.

Combined Wordline and Bitline Coding (cont.)

- W. Weeks IV and R.E. Blahut, "The capacity and coding gain of certain checkerboard codes," *IEEE Trans. Inform. Theory,* 44 (1998), 1193–1203.
- A. Kato and K. Zeger, "On the capacity of two-dimensional run length constrained channels," *IEEE Trans. Inform. Theory,* vol . 45, no. 5, July 1999, pp. 1527–1540.
- H. Ito, A. Kato, Z. Nagy, and K. Zeger, "Zero capacity region of multidimensional run length constraints," *Electr. J. Combinatorics,* 6 (1999), R33.
- S. Forchhammer and J. Justesen, "Entropy bounds for constrained two-dimensional random fields," *IEEE Trans. Inform. Theory,* 45 (1999), 118–127.

Combined Wordline and Bitline Coding (cont.)

- Z. Nagy and K. Zeger, "Capacity bounds for the 3-dimensional $(0, 1)$ runlength limited channel," *IEEE Trans. Inform. Theory*, 46 (2000), 1030–1033.
- A. Kato and K. Zeger "Partial Characterization of the Positive Capacity Region of Two-Dimensional Asymmetric Run Length Constrained Channels" *IEEE Trans. Inform. Theory*, vol. 46, no. 7, pp. 2666–2670, November 2000.
- Zs. Nagy and K. Zeger "Asymptotic Capacity of Two-dimensional Channels with Checkerboard Constraints" *IEEE Trans. Inform. Theory*, vol. 49, no. 9, pp. 2115–2125, September 2003.

Combined Wordline and Bitline Coding (cont.)

Bit-stuffing encoders

- D. E. Knuth and A. C. Yao. "The complexity of nonuniform random number generation," *Algorithms and Complexity: New Directions and Recent Results*, pp. 357–428, 1976.
- P.H. Siegel and J.K. Wolf, "Bit-stuffing bounds on the capacity of 2-dimensional constrained arrays," *ISIT'98 — IEEE Int'l Symp. Inform. Theory*, Cambridge, Massachusetts, 1998.
- R.M. Roth, P.H. Siegel, and J.K. Wolf, "Efficient Coding Schemes for the Hard-Square Model ,"*IEEE Trans. Inform. Theory*, vol. 47, no 3, pp. 1166–1176, March 2001.

Combined Wordline and Bitline Coding (cont.)

- S. Halevy, J. Chen; R.M. Roth, P.H. Siegel, J.K. Wolf, " Improved Bit-Stuffing Bounds on Two-Dimensional Constraints," *IEEE Trans. Inform. Theory,* vol. 50, no. 5, pp. 824–838, May 2004.
- Zs. Nagy and K. Zeger "Bit Stuffing Algorithms and Analysis for Run Length Constrained Channels in Two and Three Dimensions" *IEEE Trans. Inform. Theory,* vol. 50, no. 12, pp. 3146–3169, December 2004.
- I. Tal, R.M. Roth, Bounds on the rate of 2-D bit-stuffing encoders, *IEEE Trans. Inform. Theory,* 56 (2010), 2561–2567. vol. 56, no. 6, June 2010.
- I. Tal, R.M. Roth, Convex programming upper bounds on the capacity of 2-D constraints, *IEEE Trans. Inform. Theory,* 57 (2011), 381-391.

Combined Wordline and Bitline Coding (cont.)

Crossword puzzles

- C.E. Shannon, “The mathematical theory of communication,” *Bell Sys. Tech. J.*, vol. 27, pp. 379–423, 623–656, July, October 1948.
- C.E. Shannon, “Prediction and Entropy of Printed English,” *Bell Sys. Tech. J.*, vol. 30, no. 1, pp. 50–64, January 1951.
- J. K. Wolf and P.H. Siegel, “On Two-Dimensional Arrays and Crossword Puzzles,” in *Proc. 36th Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, pp. 366–371, Sept. 1998.
- Kees A.S. Immink, Paul H. Siegel, Jack K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory, Special Commemorative Issue,* vol. 44, no. 6, pp. 2260–2299, October 1998.

Acknowledgement of support

- Portions of this lecture were conceived and prepared while I was on sabbatical at Technion - Israel Institute of Technology. I thank my host Prof. Ronny Roth for useful discussions and support.

- I also gratefully acknowledge the support provided by a fellowship from the Lady Davis Foundation and by a Viterbi Leaders Fellowship in connection with my appointment as a Viterbi Visiting Faculty Chair in the Technion Computer Engineering (TCE) Center.

- Portions of my own research reported on in this lecture were supported by the National Science Foundation.