From Devices to Clouds: Coding for Modern and Next Generation Storage Systems

Ahmed Hareedy
Duke University

Seminar to the EEE Department at METU

June 4th, 2021

Introducing Myself

- **Postdoctoral Associate in the ECE Department at Duke University.**
- **Education:**
  - Ph.D. in ECE from UCLA. M.S. and Bachelor from Cairo University.
- **Industry experience:**
  - Intel Corporation and Mentor Graphics Corporation (Siemens EDA).
- **Current collaborators in academia:**

Robert Calderbank (Duke)

Siyi Yang (UCLA)

Lara Dolecek (UCLA)

Beyza Dabak (Duke)

Paul Siegel (UCSD)

Jessica Centers (Duke)

Bane Vasic (UA)

Rohith Kuditipudi (Stanford)

- **Connections in industry:**
  - Western Digital, Nokia Bell Labs, IBM Research, Intel, and Siemens.

Research Interests and Contributions

- **Research interests:**
  - Questions in coding/information theory that are fundamental to opportunities created by unparalleled access to data and computing.
  - Data storage, cloud storage, distributed computing, machine learning, DNA storage, quantum systems, and wireless communications.

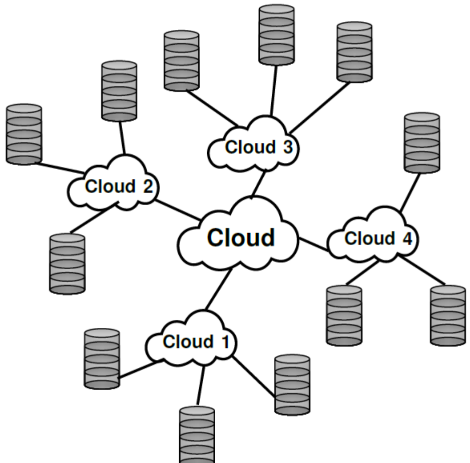
- **Research contributions:**
  - Reconfigurable constrained (LOCO) codes for storage and transmission.
  - Unequal error protection (UEP) for storage and communications.
  - Non-binary graph-based code design and optimization.
  - Spatially-coupled (SC) graph-based codes for data storage.
  - Multi-dimensional (MD) graph-based codes.
  - Performance prediction of LDPC codes over non-canonical channels.
  - Algebraic codes for flexible and scalable distributed (cloud) systems.

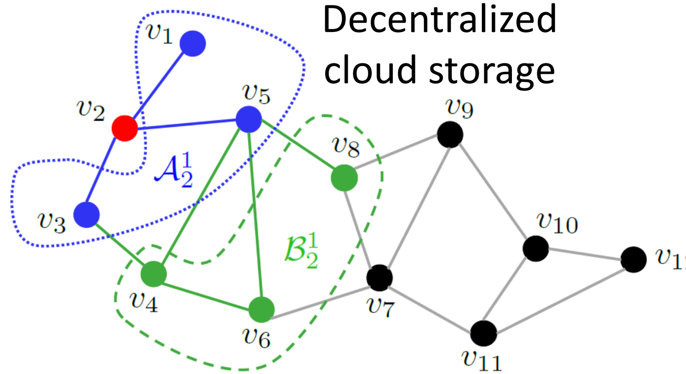
Seminar Outline

- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

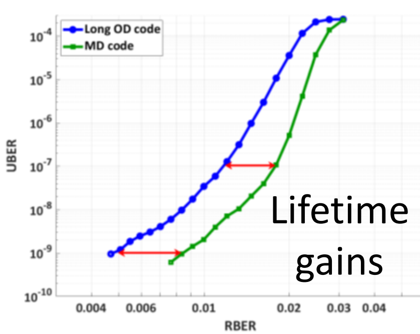
Today's Seminar in One Slide

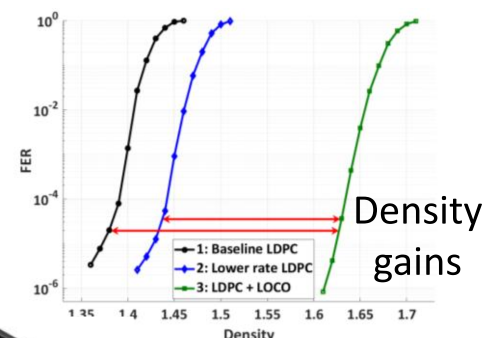
➢ **What are we going to talk about?**

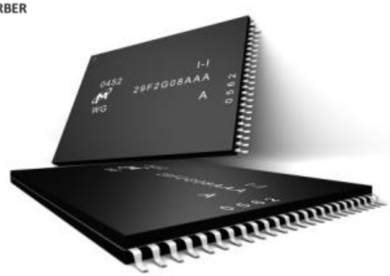




Machine learning helps coding







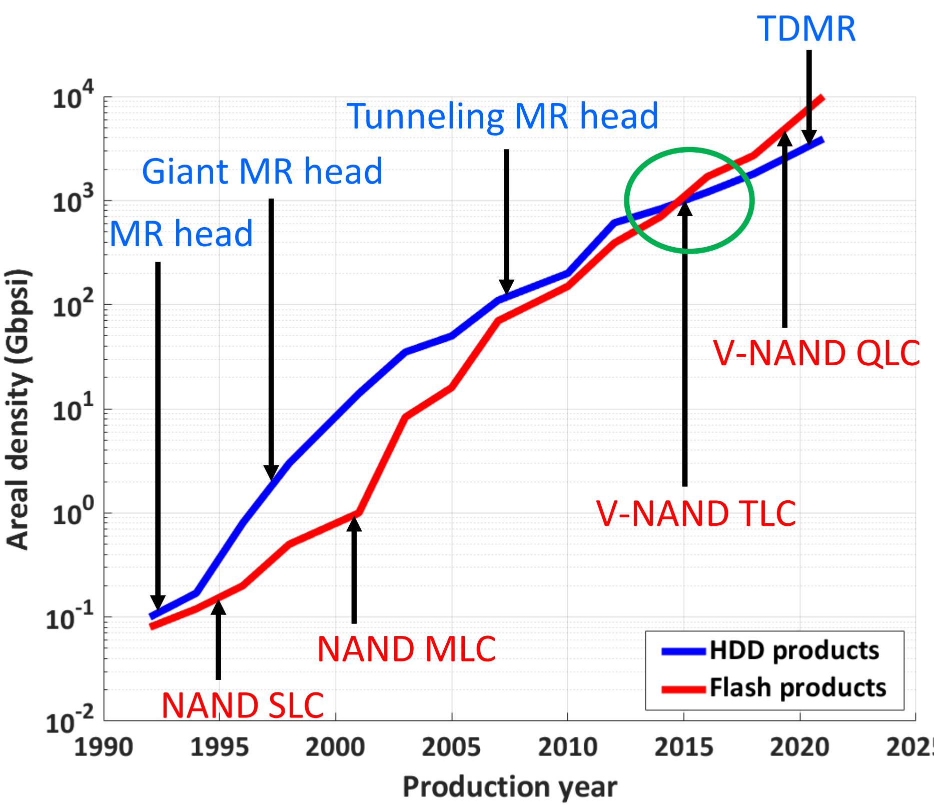
Coding helps machine learning

Storage Densities Are Rapidly Growing

- **Modern applications (IoT) require storage densities to grow rapidly.**
  - Data storage is a story where density increases as a result of advances in physics/architecture and innovations in signal processing.

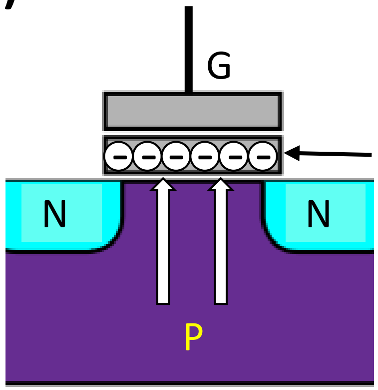
- **Data storage types:**
  - Non-volatile, magnetic (HDD).
  - Non-volatile, solid-state (Flash).
  - Non-volatile, resistive (3D XPt).
  - Volatile, solid-state (DRAM).
- **The cold-warm-hot axis.**

- **Densities approach 10 Tbpsi!**
  - With the vertical NAND (3D NAND), Flash devices are already winning!



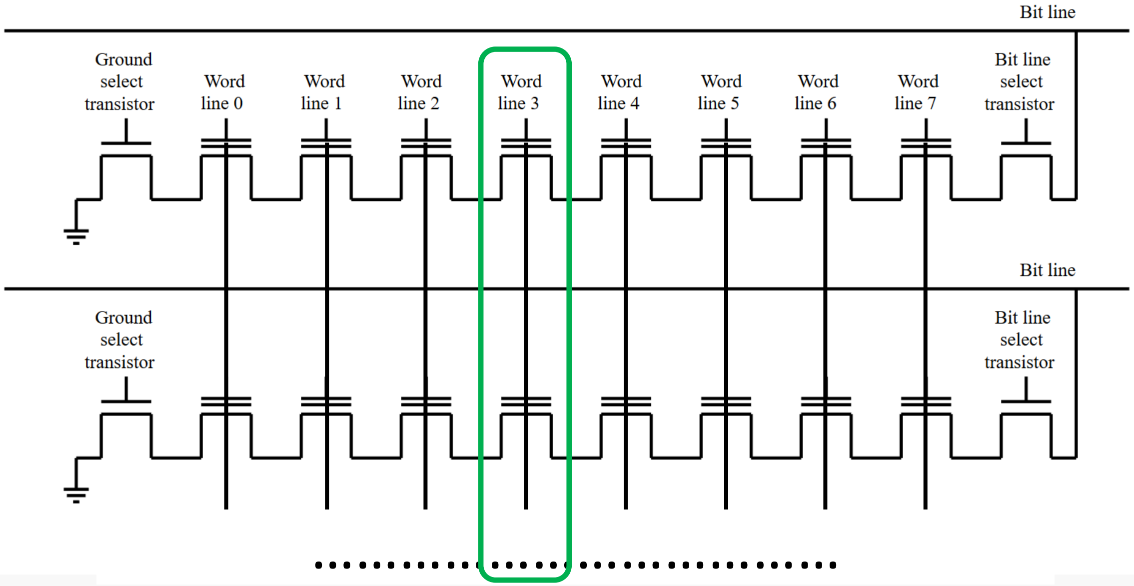
Understanding Flash Operation

- **The Flash cell is a MOSFET but with a floating gate (FG).**
  - Programming is performed via applying very high positive voltage to the gate (NPN).
  - Electrons tunnel into the FG.
  - The charge level in the FG controls threshold.



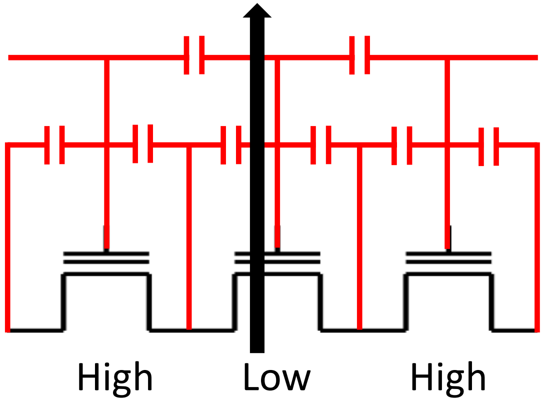
- **Advances in physics enabled more than two charge levels per cell (SLC vs. M/T/Q/P-LC).**

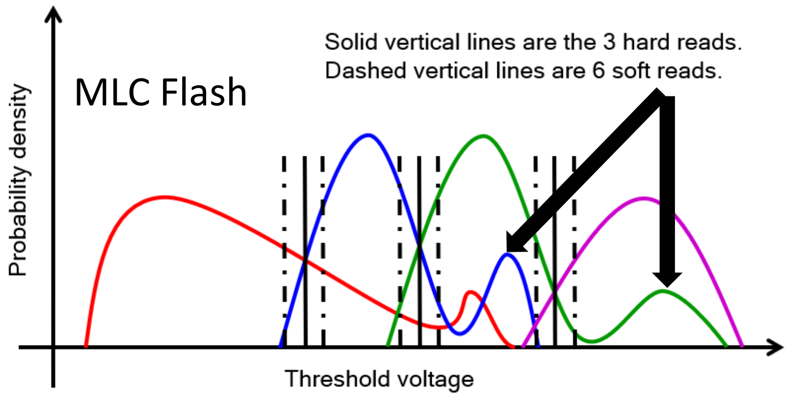
- **How to read Word 3 in NAND Flash:**
  - Apply ON voltage to all word lines except 3, and read voltage to 3.



Sources of Error in Flash Devices

- **Inter-cell interference (ICI):**
  - Parasitic capacitances result in charge propagation (101 in SLC).
- **Programming (wear-out) errors:**
  - Failed programming/erasing operations result in asymmetric errors.
- **Other sources of error:**
  - Charge leakage over time and Gaussian electronic noise.





- **What about magnetic recording devices?**
  - Inter-symbol interference (ISI), inter-track interference (in TDMR), jitter or timing problems, and Gaussian noise.

How Can We Take Full Advantage?

- **Data storage devices operate at very low error rates.**
- **My technical vision is:**
  - To devise efficient coding techniques that exploit the advances in physics to significantly improve performance.

- **Mitigating interference:**
  - Constrained codes prevent error-prone patterns from being written.
  - **LOCO codes** forbid these patterns with minimal redundancy.
  - LOCO codes can be easily reconfigured as the device ages.

- **Handling other sources of error:**
  - Graph-based (LDPC) codes correct the errors after reading.
  - **OO/GRADE-AO techniques** generate powerful custom SC codes.
  - Careful coupling of SC codes generates excellent MD codes.

- **These techniques result in significant lifetime and density gains!**

Seminar Outline

- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

Introduction to Constrained Codes

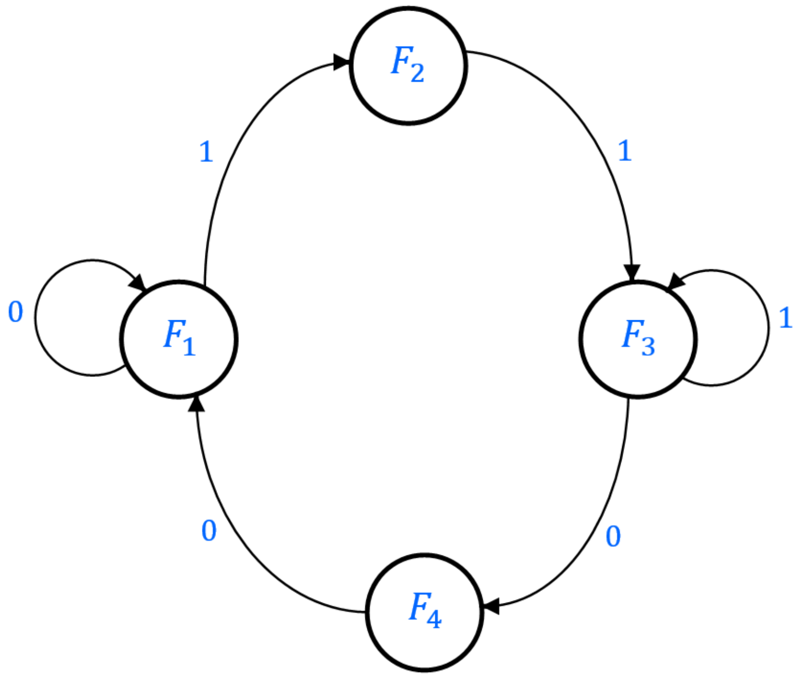
- **Constrained codes impose restrictions on written (transmitted) data.**
  - The set of forbidden patterns can be symmetric or asymmetric.
  - The rate is (# of input bits)/(# of coded bits or symbols).

- **The universe of constrained sequences is represented by an FSTD. The capacity, i.e., the highest achievable rate, is the graph entropy.**

- **Example:** $\mathcal{S}_1 = \{010, 101\}$ constraint.
  - The adjacency matrix of this FSTD is:

$$\mathbf{F} = \begin{array}{c} \\ F_1 \\ F_2 \\ F_3 \\ F_4 \end{array} \begin{array}{c} \begin{array}{cccc} F_1 & F_2 & F_3 & F_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

  - The capacity is $\log_2(\lambda_{\max}(\mathbf{F}))$, which is 0.6942 here.



History and My LOCO Codes

Franaszek

1948 Shannon: FSTD of constrained codes [R1]

1970 Tang and Bahl: RLL codes [R2]

Cover: Enumerative source coding [R3]

1980 Adler, Coppersmith, and Hassner: State splitting and merging to produce FSMs [R4]

1990 Karabed, Immink, Siegel, and Wolf: Optimization of FSMs [R5, R6]

2000 Immink and Braun: Constrained codes based on lexicographic indexing [R7, R8]

2020 LOCO codes: Constrained codes that are capacity-achieving, simple, and reconfigurable [H1, H2]

General method for finite sets

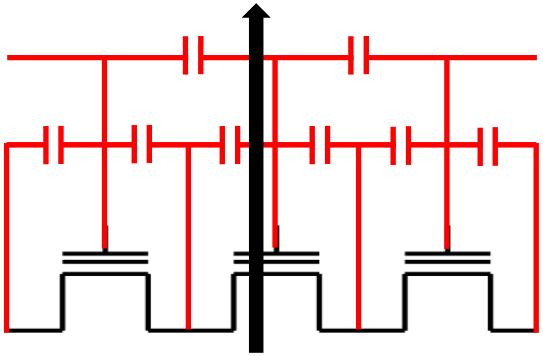
Claude Shannon

Kees Immink

Device Physics Determine Patterns to Forbid

- **Consider Flash devices with q levels per cell:**
  - SLC ($q = 2$), MLC ($q = 4$), TLC ($q = 8$), QLC ($q = 16$), PLC ($q = 32$).
  - Symbols in GF(q) $= \{0, 1, \alpha, \dots, \alpha^{q-2}\}$ are written as charge (threshold) levels in $\{0, 1, 2, \dots, q-1\}$.

- **What should we forbid?**
  - Patterns resulting in max charge at the outer cells but less at the inner ones [R9].
  - Let δ be in GF(q)$\backslash\{\alpha^{q-2}\}$. The set of forbidden patterns is:
    $$\mathcal{Q}_x^q \triangleq \left\{\alpha^{q-2}\boldsymbol{\delta}_{\mathrm{d}}^{y}\alpha^{q-2}, \forall \boldsymbol{\delta}_{\mathrm{d}}^{y} \in [\mathrm{GF}(q)\backslash\{\alpha^{q-2}\}]^{y} \mid 1 \le y \le x\right\}.$$
  - If $q = 2$ (binary), $\mathcal{Q}_x^2 = \{101, 1\mathbf{0}^2 1, \dots, 1\mathbf{0}^x 1\}$.



  - The codes are q-ary asymmetric LOCO (QA-LOCO) codes.
  - Handling $x > 1$ can increase the lifetime and reduce the time to market.

Formal Definition and Group Structure

- **A QA-LOCO code $\mathcal{QC}_{m,x}^{q}$ is defined by:**
  - Each codeword $\mathbf{c} \in \mathcal{QC}_{m,x}^{q}$ has symbols in GF(q) and is of length m.
  - Codewords in $\mathcal{QC}_{m,x}^{q}$ are ordered lexicographically.
  - Each codeword $\mathbf{c} \in \mathcal{QC}_{m,x}^{q}$ does not contain any pattern in $\mathcal{Q}_{x}^{q}$, $x \geq 1$.
  - All codewords satisfying the above properties are included.

- **Codewords in $\mathcal{QC}_{m,x}^{q}$, $m \geq 2$, are partitioned into three groups:**
  - Group 1: Codewords starting with δ, $\forall \delta$, from the left.
  - Group 2: Codewords starting with $\alpha^{q-2}\alpha^{q-2}$ from the left.
  - Group 3: Codewords starting with $\alpha^{q-2}\boldsymbol{\delta}_{\mathrm{d}}^{x+1}$, $\forall \boldsymbol{\delta}_{\mathrm{d}}^{x+1}$, from the left.

- **What QA-LOCO codes offer [H3]:**
  - They mitigate ICI, and they are capacity-achieving.
  - They have simple encoding-decoding, and they are reconfigurable.

QA-LOCO Codes With $q = 2$ and $x = 1$

Index $g(\mathbf{c})$	Codewords of the code $\mathcal{QC}_{m,1}^{2}$				
	$m = 1$	$m = 2$	$m = 3$	$m = 4$	
0	0	00	000	0000	Group 1
1	1	01	001	0001	
2		10	010	0010	
3		11	011	0011	
4			100	0100	
5			110	0110	
6			111	0111	
7				1000	Group 3
8				1001	
9				1100	Group 2
10				1110	
11				1111	

Enumerating the Codewords

- **Theorem:** Let $N_q(m,x)$ be the cardinality of $\mathcal{QC}^q_{m,x}$. Define:

$$N_q(m,x) \triangleq (q-1)^m, -x \leq m \leq 0, \text{ and } N_q(1,x) \triangleq q.$$

Then, $N_q(m,x), m \geq 2$, is recursively given by:

$$N_q(m,x) = qN_q(m-1,x) - (q-1)N_q(m-2,x)$$
$$+(q-1)^{x+1}N_q(m-x-2,x).$$

- **Example:** For $q = 2$ and $x = 1$:

$$N_2(m,1) = 2N_2(m-1,1) - N_2(m-2,1) + N_2(m-3,1).$$

  - $N_2(-1,1) \triangleq 1, N_2(0,1) \triangleq 1, N_2(1,1) \triangleq 2.$
  - $N_2(2,1) = 2N_2(1,1) - N_2(0,1) + N_2(-1,1) = 4.$
  - $N_2(3,1) = 2N_2(2,1) - N_2(1,1) + N_2(0,1) = 7.$
  - $N_2(4,1) = 2N_2(3,1) - N_2(2,1) + N_2(1,1) = 12.$
  - The numbers are consistent with the table.

Encoding-Decoding Rule of the Codes

- **Theorem:** The index of a QA-LOCO codeword $\mathbf{c} \triangleq c_{m-1}c_{m-2} \dots c_0 \in \mathcal{QC}^{q}_{m,x}$, $m \geq 2$, is given by the rule:

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} a_i (q-1)^{\gamma_i} N_q(i-\gamma_i, x),$$

where $a_i \triangleq \text{gflog}_\alpha(c_i) + 1$, $c_i \neq 0$, is the level equivalent of c_i, and $\gamma_i = x - k_i + 1$; k_i is the distance to the closest α^{q-2} symbol.

  - For example, if $c_6c_5 = \alpha^{q-2}\alpha$, then $a_6 = q-1$, $k_5 = 1$, and $\gamma_5 = x$.

- **For the binary case ($q = 2$):**

$$g(\mathbf{c}) = \sum_{i=0}^{m-1} a_i N_2(i - a_{i+1}x, x).$$

- **Example:** $q = 2$, $m = 4$, and $x = 1$:
  - $g(\mathbf{c} = 1110)$
  $= \sum_{i=0}^{3} a_i N_2(i - a_{i+1}, 1)$
  $= N_2(3,1) + N_2(1,1) + N_2(0,1)$
  $= 7 + 2 + 1 = 10.$

Index $g(\mathbf{c})$	Codewords of the code $\mathcal{QC}^{2}_{4,1}$	
0	0000	Group 1
1	0001	
2	0010	
…	…………	
7	1000	Group 3
8	1001	
9	1100	Group 2
10	1110	
11	1111	

Data Protection Almost for Free!

- **Bridging is needed to prevent forbidden patterns across codewords.**
  - Bridge with x consecutive 0's or x consecutive α^{q-2}'s.
- **Self-clocking is needed to maintain calibration of the system.**
  - Just remove the all 0 and the all α^{q-2} codewords from $\mathcal{QC}_{m,x}^{q}$.
- **The rate of a self-clocked QA-LOCO code in input bits/coded symbol is:**

$$R_{\mathrm{QA-LOCO}}^{\mathrm{c}} = \frac{s^{\mathrm{c}}}{m+x} = \frac{\left\lfloor \log_2\left(N_q(m,x) - 2\right)\right\rfloor}{m+x}.$$

  - Codes are capacity-achieving.
- **Rate examples for $x = 1$:**
  - **Exploiting physics:** Less than 3% redundancy suffices for ICI mitigation!
  - Achieved at low complexity.

$q = 8$ (TLC)		$q = 16$ (QLC)	
m	Norm rate	m	Norm rate
26	0.9506	27	0.9554
44	0.9704	45	0.9728
71	0.9769	66	0.9813
Capacity	0.9939	Capacity	0.9987

Reconfigurability and Comparisons

- **Encoding and decoding of QA-LOCO codes are performed via the rule.**
  - Encoding: Mapping from index to codeword (subtractions).
  - Decoding: Demapping from codeword to index (additions).
- **The same hardware can support multiple constraints by updating N's.**

- **QA-LOCO codes can be easily reconfigured [H3].**
  - As the device ages, the set of patterns to forbid becomes bigger ($x > 1$).
  - Reconfiguration is as easy as reprogramming an adder!
  - A small number of multiplexers pick the appropriate cardinalities.

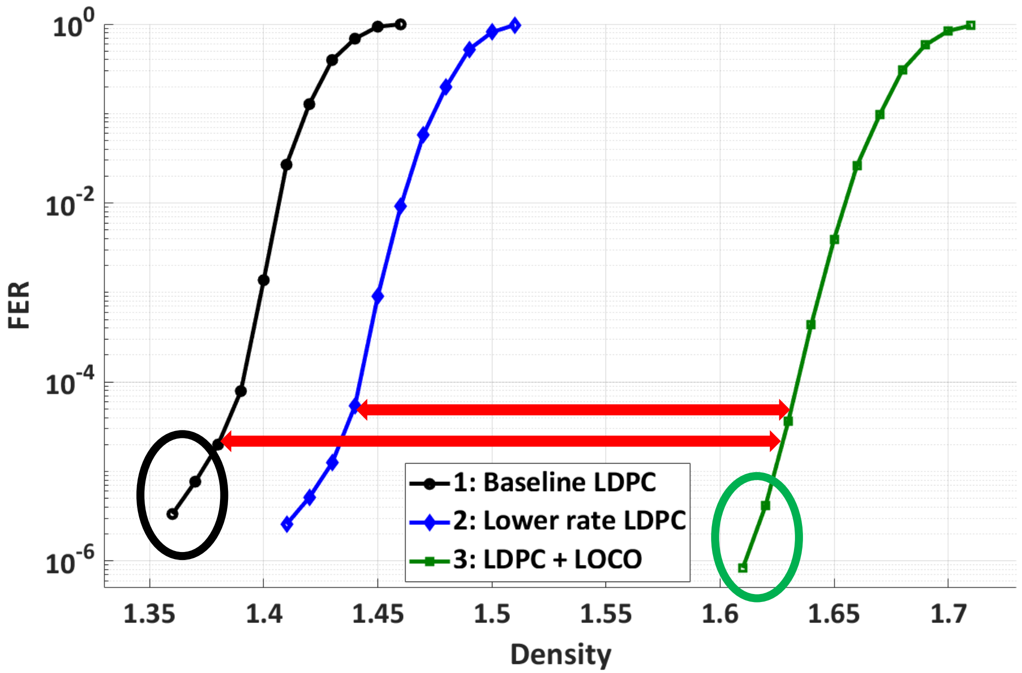
- **Comparisons vs. other techniques:**

Near-optimal LOCO solutions can help

  - It is quite complicated to design capacity-achieving non-binary constrained codes based on FSMs.
  - Other codes either do not exploit Flash physics [R2], incur higher complexity [R10], or designed only for $x = 1$ [R10, R11].

UEP Achieves Significant Density Gains

- **I simulated three setups in an industry-recommended MR system.**
- **Setup 3 (LDPC + LOCO on parity bits only) achieves [H1]:**
  - About 20% (16%) density gain compared with Setup 1 (Setup 2).
  - Investing the additional redundancy via LOCO is more beneficial!
  - Even the error floor performance in Setup 3 is better.
  - I theoretically demonstrated such UEP gains on canonical channels [H4].



Overall length 4270 bits
Overall rate 0.645

The diffusion of more reliable information provides the LDPC decoder with a better channel.

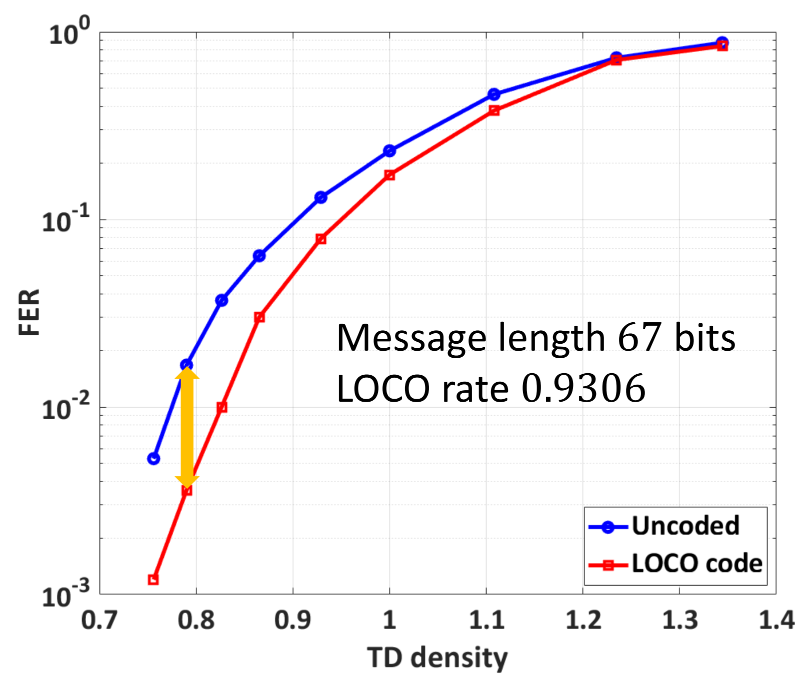
Constrained Codes for TDMR

- **The TDMR technology does not require new magnetic materials.**
  - Shingled writing, squeezed tracks, and advanced signal processing are adopted to remarkably increase MR densities [H2].

- **With wide read heads, error-prone patterns become two-dimensional.**
  - They take the shape of a plus sign (+): Plus isolation patterns.

·	0	·
0	1	0
·	0	·

·	1	·
1	0	1
·	1	·



- **LOCO codes achieve significant performance gains in TDMR even before LDPC decoding [H2].**

My Related Work (1 of 2)

- [H1] **A. Hareedy** and R. Calderbank, "LOCO codes: Lexicographically-ordered constrained codes," *IEEE Trans. Inf. Theory*, 2020.

- [H2] **A. Hareedy**, B. Dabak, and R. Calderbank, "The secret arithmetic of patterns: A general method for designing constrained codes based on lexicographic indexing," *ArXiv* and submitted to *IEEE Trans. Inf. Theory*, 2020.

- [H3] **A. Hareedy**, B. Dabak, and R. Calderbank, "Managing device lifecycle: Reconfigurable constrained codes for M/T/Q/P-LC Flash memories," *IEEE Trans. Inf. Theory*, 2021.

- [H4] B. Dabak, **A. Hareedy**, A. Ashikhmin, and R. Calderbank, "Unequal error protection achieves threshold gains on BEC and BSC via higher fidelity messages," *ArXiv*, 2021.

- B. Dabak, **A. Hareedy**, and R. Calderbank, "Non-binary constrained codes for two-dimensional magnetic recording," *IEEE Trans. Magn.*, 2020.

My Related Work (2 of 2)

- J. Centers, X. Tan, **A. Hareedy**, and R. Calderbank, "Power spectra of constrained codes with level-based signaling: Overcoming finite-length challenges," *IEEE Trans. Commun.*, 2021.
- **A. Hareedy** and R. Calderbank, "A new family of constrained codes with applications in data storage," in *Proc. IEEE ITW*, 2019.
- **A. Hareedy** and R. Calderbank, "Asymmetric LOCO codes: Constrained codes for Flash memories," in *Proc. Allerton*, 2019.
- **A. Hareedy**, B. Dabak, and R. Calderbank, "Q-ary asymmetric LOCO codes: Constrained codes supporting Flash evolution," in *Proc. IEEE ISIT*, 2020.

References (1 of 2)

- [R1] C. E. Shannon, "A mathematical theory of communication," Bell Sys. Tech. J., 1948.
- [R2] D. T. Tang and R. L. Bahl, "Block codes for a class of constrained noiseless channels," *Inf. and Control*, 1970.
- [R3] T. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, 1973.
- [R4] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes–An application of symbolic dynamics to information theory," *IEEE Trans. Inf. Theory*, 1983.
- [R5] R. Karabed and P. H. Siegel, "Coding for higher-order partial-response channels," in *Proc. SPIE 2605,* 1995.
- [R6] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, "Codes for digital recorders," *IEEE Trans. Inf. Theory*, 1998.

References (2 of 2)

- [R7] K. A. S. Immink, “A practical method for approaching the channel capacity of constrained channels,” *IEEE Trans. Inf. Theory*, 1997.
- [R8] V. Braun and K. A. S. Immink, “An enumerative coding technique for DC-free runlength-limited sequences,” *IEEE Trans. Commun.*, 2000.
- [R9] 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*, 2015.
- [R10] Y. M. Chee, J. Chrisnata, H. M. Kiah, S. Ling, T. T. Nguyen, and V. K. Vu, “Capacity-achieving codes that mitigate intercell interference and charge leakage in Flash memories,” *IEEE Trans. Inf. Theory*, 2019.
- [R11] M. Qin, E. Yaakobi, and P. H. Siegel, “Constrained codes that mitigate inter-cell interference in read/write cycles for flash memories,” *IEEE J. Sel. Areas Commun.*, 2014.

Seminar Outline

- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

Introduction to LDPC Codes

- **Parity-check codes are a class of block error-correction codes (ECCs).**
  - The code is defined by a parity-check matrix $\mathbf{H}$.
  - A codeword $\mathbf{v}$ satisfies $\mathbf{H}\mathbf{v}^{\mathrm{T}} = \mathbf{0}$.

Systematic form

$$\mathbf{H}_{(n-k)\times n} = \left[\mathbf{P}^{\mathrm{T}}_{(n-k)\times k}\ \mathbf{I}_{(n-k)\times(n-k)}\right], \mathbf{G}_{k\times n} = \left[\mathbf{I}_{k\times k}\ \mathbf{P}_{k\times(n-k)}\right].$$

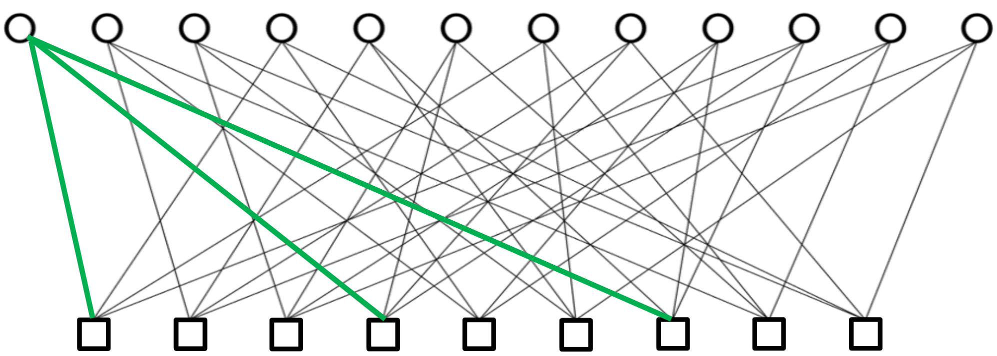
Robert Gallager

$\mathbf{H} =$

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

Columns represent bit or variable nodes (VNs).
Rows represent check nodes (CNs).
Non-zero values represent edges.

The corresponding bipartite graph:
Circles represent VNs.
Squares represent CNs.



Message Passing and Lifting

- **Decoding is iterative; via messages between VNs and CNs [R12].**

$$c_1 = v_1 \oplus v_2 \oplus v_3 \oplus v_5,$$
$$c_2 = v_1 \oplus v_2 \oplus v_4 \oplus v_6,$$
$$c_3 = v_1 \oplus v_3 \oplus v_4 \oplus v_7,$$
$$c_4 = v_5 \oplus v_6 \oplus v_7.$$

$$\mathbf{H} = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}.$$

  - Binary example: Gallager A decoding, and we receive $[0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0]$.
  - CNs c_1 and c_2 are the only unsatisfied CNs. VN v_2 flips to 1 for $\mathbf{H}\mathbf{v}^{\mathrm{T}} = 0$.

- **Lifting a protograph (seed) to generate an LDPC code:**
  - γ (κ) is the column (row) weight, i.e., VN (CN) degree.
  - $\mathbf{H}^{\mathrm{p}}$ is the protograph matrix. σ is the $z \times z$ circulant matrix, $\sigma^0 = \mathbf{I}$.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$\mathbf{H}^{\mathrm{p}}$ with $\gamma = 3$ and $\kappa = 5$

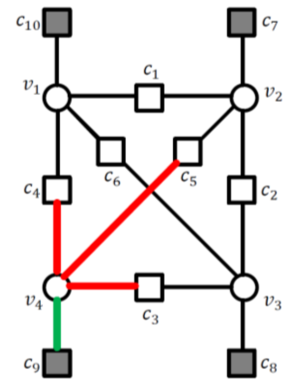
Lifting with $z = 5$

σ^3	σ^2	σ^0	σ^4	σ^1
σ^1	σ^0	σ^2	σ^3	σ^4
σ^1	σ^2	σ^4	σ^0	σ^3

$\mathbf{H}$

Detrimental Objects in LDPC Codes

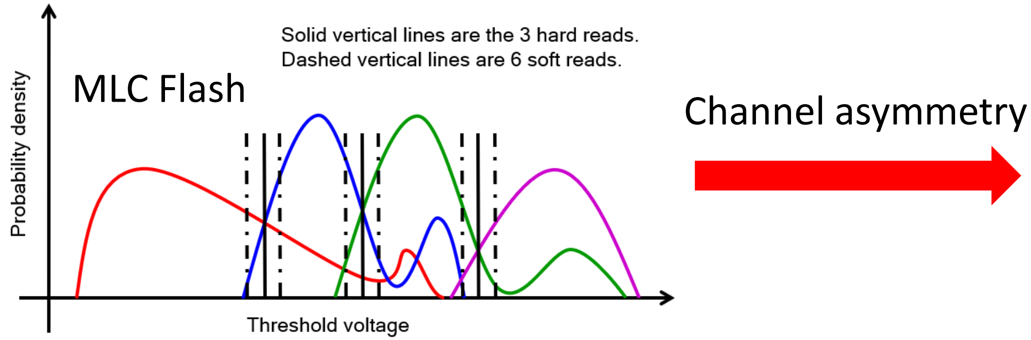
- **Absorbing sets [R13, R14] result in decoding failure → error floor.**
  - For an (a, b) absorbing set: The size of the set is a, the number of unsatisfied CNs connected to it is b, and each VN is connected to more satisfied than unsatisfied neighboring CNs.
  - A $(4, 4)$ binary absorbing set ($\gamma = 4$):
  - More parameters are added for non-binary.

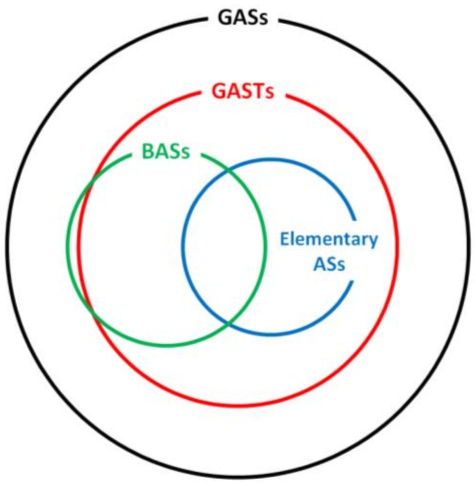


Circles represent VNs. White (grey) squares represent satisfied (unsatisfied) CNs.

- **Define an (a, d_1) unlabeled elementary trapping (absorbing) set (UTS) ((UAS)).**

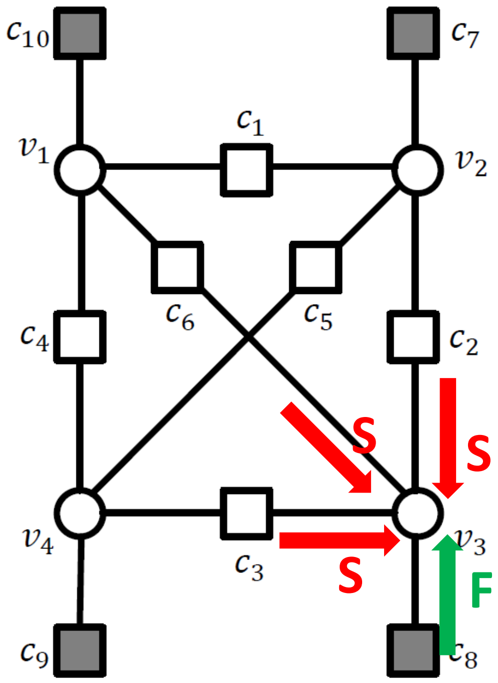
- **Detrimental objects depend on the physics [H5].**





Absorbing Sets Absorb the Decoder

- **How can an absorbing set (AS) cause a decoding error?**
  - Assume the all 0 codeword is transmitted.
  - Assume errors occur only on the four VNs in the shown $(4,4)$ AS in the graph of the binary LDPC code.
  - Thus, all these VNs are now 1's.
  - Consider hard decision decoding.
  - Each degree-2 CN now is satisfied $(1+1=0)$, while degree-1 CNs are not.
  - Each VN receives 3 stay and only 1 flip messages from the connected CNs.
  - Despite being in error, all VNs stick to their wrong values.
  - Consequently, the decoder is **absorbed**!

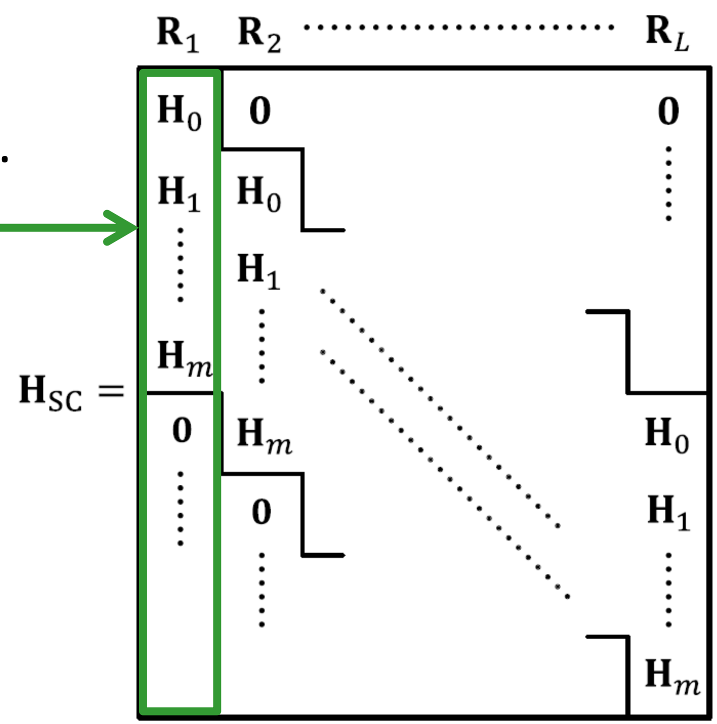


Construction of SC Codes

- **SC codes have excellent error-correction performance [R15].**
  - They offer additional degrees of freedom in the code design.

- **The construction steps are:**
  - Partition $\mathbf{H}$ (size $\gamma z \times \kappa z$) into $m+1$ components: $\mathbf{H}_0, \mathbf{H}_1, \ldots, \mathbf{H}_m$.
  - Couple component matrices L times to construct $\mathbf{H}_{\text{SC}}$ (size $\gamma z(L+m) \times \kappa z L$).
  - If non-binary, assign weights $\in$ GF$(q)\backslash\{0\}$.

$$\mathbf{H} \triangleq \sum_{y=0}^{m} \mathbf{H}_y,\ \mathbf{H}^{\mathrm{p}} \triangleq \sum_{y=0}^{m} \mathbf{H}_y^{\mathrm{p}} \text{ (all 1's).}$$



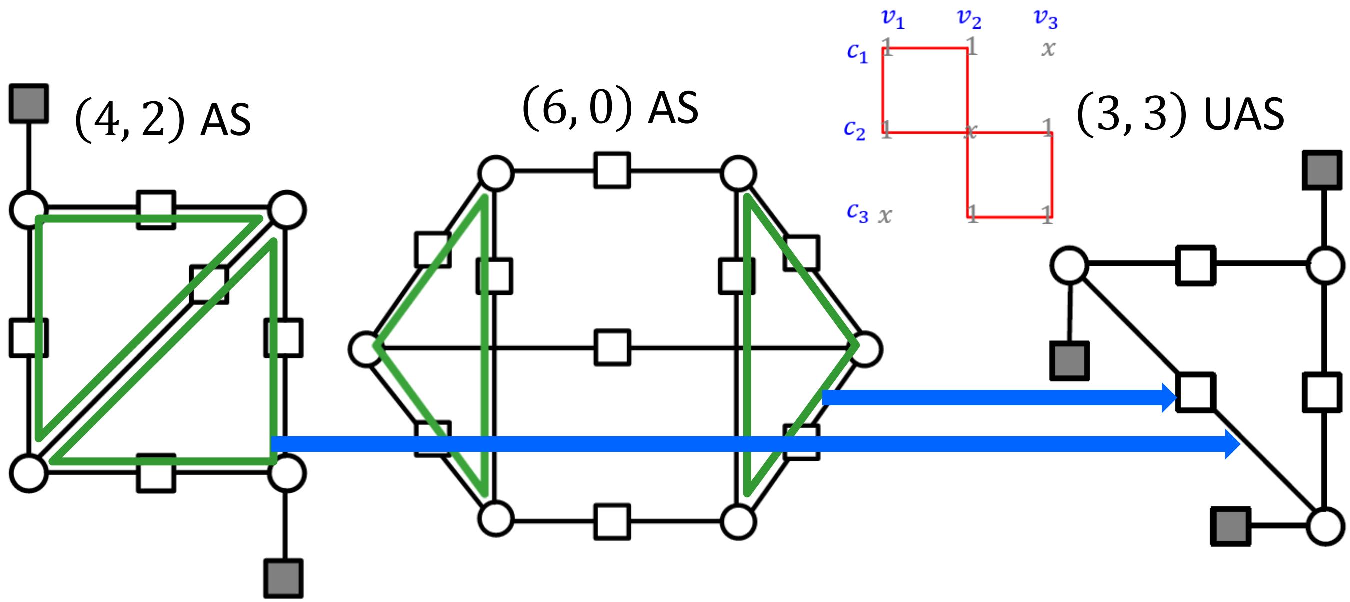
- **My goal is to eliminate detrimental objects via optimized partitioning and lifting.**
  - We know such objects in data storage systems (differ from AWGN) [H6].

What Techniques Do I Propose?

- **Previous work on partitioning includes [R16], [R17], and [H7].**

- **Operate on the protograph then the unlabeled graph to design $\mathbf{H}_{\text{SC}}$:**
  - For low m, derive the optimal partitioning (OO) [H8, H9].
  - For high m, derive a near-optimal partitioning (GRADE-AO) [H10].
  - Next, optimize the lifting (CPO) [H8, H9]. Stop here if binary (the focus).
  - If non-binary, optimize the edge weights (WCM) [H5, H6].

- **Common substructures:**
  - Minimize # of cycles-6 and cycles-8.



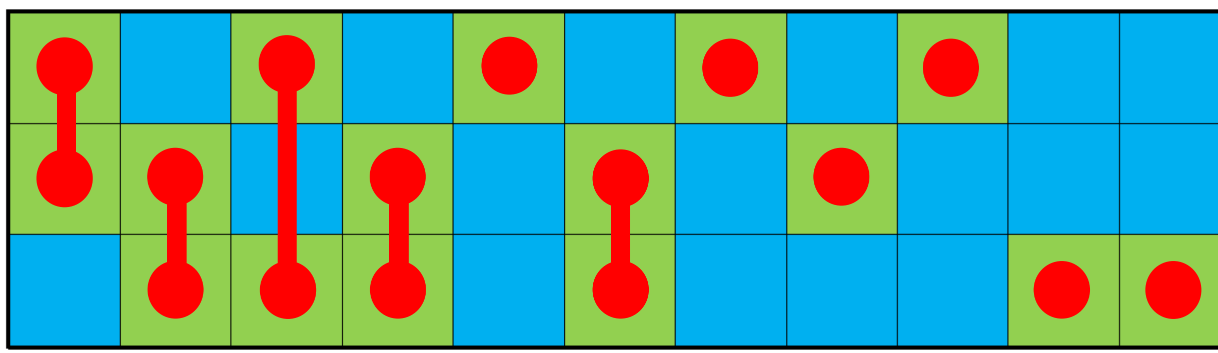
OO: What Are the Overlap Parameters?

- **The set of** **independent non-zero overlap parameters** **is** $\mathcal{O}_{\text{ind}}$**.**
- **<u>Example:</u>** For $\gamma = 3$ and $m = 1$:

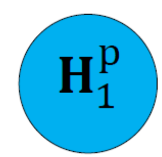
  $\mathcal{O}_{\text{ind}} = \{t_{\{0\}}, t_{\{1\}}, t_{\{2\}}, t_{\{0,1\}}, t_{\{0,2\}}, t_{\{1,2\}}, t_{\{0,1,2\}}\}$ (the ones in $\mathbf{H}_0^{\text{p}}$).
  - Other overlap parameters are functions of the ones in $\mathcal{O}_{\text{ind}}$.

- **I illustrate their definitions via an example:**
  - Consider the case of $\kappa = 11$:
  - $t_{\{0\}} = 5$.
  - $t_{\{1\}} = 5$.
  - $t_{\{2\}} = 6$.
  - $t_{\{0,1\}} = 1$.
  - $t_{\{0,2\}} = 1$.
  - $t_{\{1,2\}} = 3$.
  - $t_{\{0,1,2\}} = 0$.

$\mathbf{H}^{\text{p}} =$



Define a degree-μ overlap and a degree-μ overlap parameter.



Building a Discrete Optimization Problem

- **<u>Theorem:</u>** The total number of cycle-6 instances in the protograph of an SC code with $\gamma \geq 3$, κ, m, and $L \geq m+1$ is:

$$F = \sum_{k=1}^{m+1} (L-k+1) F_1^k,$$

$$\begin{aligned} &\mathcal{A}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_2,i_3\}}, t_{\{i_1,i_2,i_3\}}\right) \\ &\quad = t_{\{i_1,i_2,i_3\}} \left(t_{\{i_1,i_2,i_3\}} - 1\right)^+ \left(t_{\{i_2,i_3\}} - 2\right)^+ \\ &\quad + t_{\{i_1,i_2,i_3\}} \left(t_{\{i_1,i_3\}} - t_{\{i_1,i_2,i_3\}}\right) \left(t_{\{i_2,i_3\}} - 1\right)^+ \\ &\quad + \left(t_{\{i_1,i_2\}} - t_{\{i_1,i_2,i_3\}}\right) t_{\{i_1,i_2,i_3\}} \left(t_{\{i_2,i_3\}} - 1\right)^+ \\ &\quad + \left(t_{\{i_1,i_2\}} - t_{\{i_1,i_2,i_3\}}\right) \left(t_{\{i_1,i_3\}} - t_{\{i_1,i_2,i_3\}}\right) t_{\{i_2,i_3\}}. \end{aligned}$$

  F_1^k is the number of instances starting from $\mathbf{R}_1$ and spanning k replicas.

$$F_1^1 = \sum_{\{i_1,i_2,i_3\} \subset \{0,\dots,(m+1)\gamma-1\}} \mathcal{A}\left(t_{\{i_1,i_2\}}, t_{\{i_1,i_3\}}, t_{\{i_2,i_3\}}, t_{\{i_1,i_2,i_3\}}\right),$$

  with $\overline{i_1} \neq \overline{i_2}$, $\overline{i_1} \neq \overline{i_3}$, $\overline{i_2} \neq \overline{i_3}$, and $\overline{i_x} \triangleq (i_x \bmod \gamma)$.

- **The discrete optimization problem is described as follows.**
  - <u>Mathematical formulation:</u>

$$F^* = \min_{\mathcal{O}_{\text{ind}}} F.$$

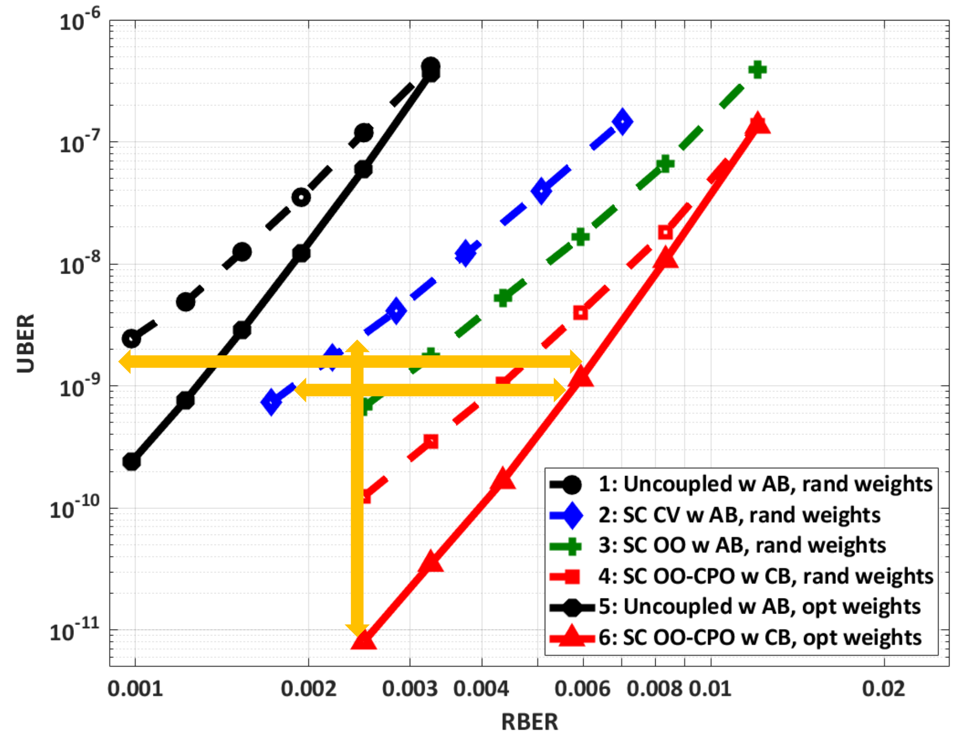
  - <u>Optimization constraints:</u>

    Interval constraints and the balanced (uniform) partitioning constraint.
  - A solution to this problem is $\mathbf{t}^*$. $\mathbf{t}^*$ is called an **optimal vector**.

- **The CPO then breaks the reflection condition [R18] for as many cycles in the optimal SC protograph (designed via $\mathbf{t}^*$) as possible.**

Notable Performance Gains in Flash

- **Channel: Normal-Laplace mixture (NLM) Flash [R19].**
  - IBM MLC channel, with 3 reads and sector size 512 bytes.
  - RBER is raw BER. UBER is uncorrectable BER $= \text{FER}/(512 \times 8)$.
- **All the codes have $\gamma = 3$, $\kappa = z = 19$, $m = 1$, $L = 20$, and $q = 4$.**
  - Length 14440 bits and rate 0.834.



- **The OO-CPO-WCM approach outperforms existing methods:**
  - Code 6 outperforms Code 2 by 2.5 orders of magnitude.
  - Code 6 achieves 200% RBER gain compared with Code 2.
- **Appropriately-designed SC codes outperform block codes [H9].**

GRADE-AO: Gradient Descent Optimizer

- **SC codes perform better as the memory becomes higher.**
  - The complexity of the OO technique grows rapidly with m and γ.

- **GRADE-AO is a probabilistic technique that enables high memories.**
  - Denote the probability (edge) distribution by $\mathbf{p} \triangleq [p_0\ p_1\ \dots\ p_m]$.
  - Define the polynomial $f(X, \mathbf{p}) \triangleq \sum_{i=0}^{m} p_i X^i$. $[\cdot]_i$ is the coefficient of X^i.

- **Theorem:** A necessary condition to minimize the probability of a cycle-6 under random partitioning is $[f^3(X, \mathbf{p}) f^2(X^{-1}, \mathbf{p})]_i = c_0, \forall i \in \{0, 1, \dots, m\}$.
  - This probability is $[f^3(X, \mathbf{p}) f^3(X^{-1}, \mathbf{p})]_0$.
  - Consider $\mathcal{L}_6(\mathbf{p}) = [f^3(X, \mathbf{p}) f^3(X^{-1}, \mathbf{p})]_0 + c[1 - \sum_{i=0}^{m} p_i]$.
  - Then, $\nabla_{\mathbf{p}} \mathcal{L}_6(\mathbf{p}) = 0$ leads to the necessary condition.

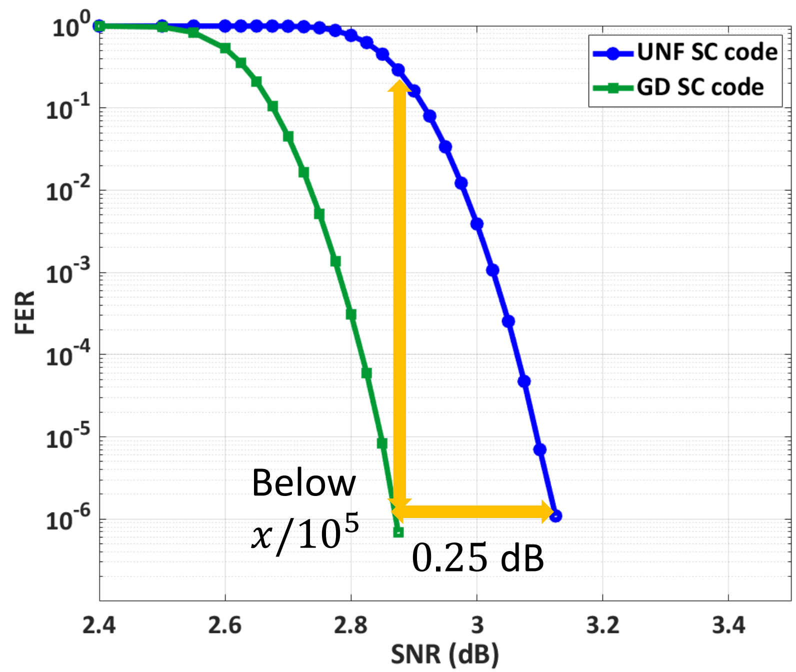
- **Gradient descent is then used to find $\mathbf{p}$ that satisfies the condition.**
  - GRADE-AO plus CPO give $\mathbf{H}_{\mathrm{SC}}$. Analysis is also done for cycles-8 [H10].

Uniform Partitioning Is Not the Answer!

- **Now, compare the gradient descent (GD) SC codes with high performance uniform (UNF) SC codes.**

$(\gamma, \kappa, z, m, L)$	Code	Cycles-6	Cycles-8
$(3, 7, 13, 5, 100)$	GD	0	0
	UNF	0	6292
$(4, 29, 29, 19, 20)$	GD	0	528090
	UNF	0	1087268

Performance of $(4, 29, 29, 19, 20)$ SC codes over the AWGN channel



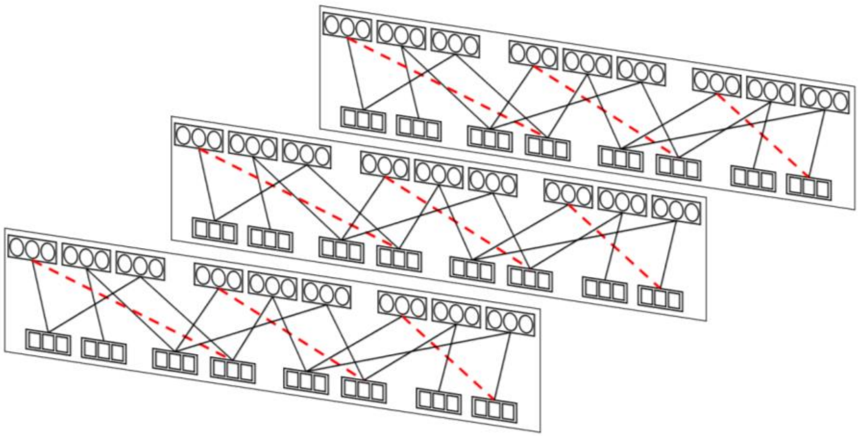
- **GD SC codes have superior performance in all regions.**
  - They have potential in data storage and wireless communication systems.

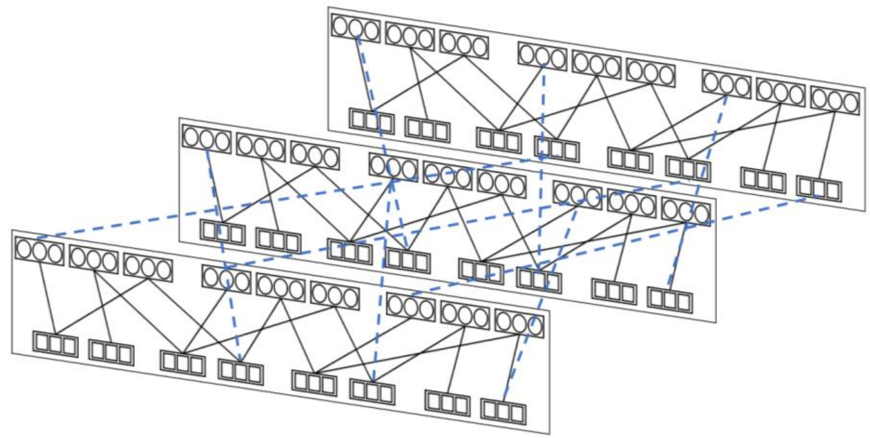
- **For high m, vastly skewed distributions give better thresholds!**
  - We are working on theoretical justification.

Alexei Ashikhmin

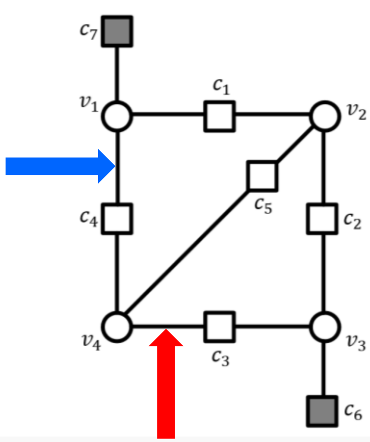
Construction of MD Graph-Based Codes

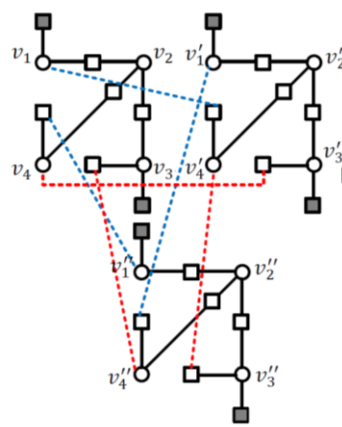
- **What is the idea of my technique?**
  - Optimally couple multiple copies of a high performance OD code to mitigate (MD) system non-uniformity [R20] in storage devices.

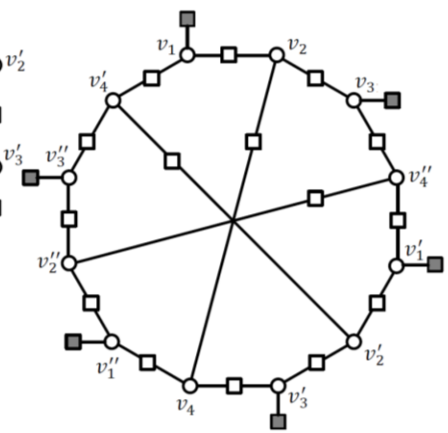




- **Effective MD coupling removes detrimental objects.**

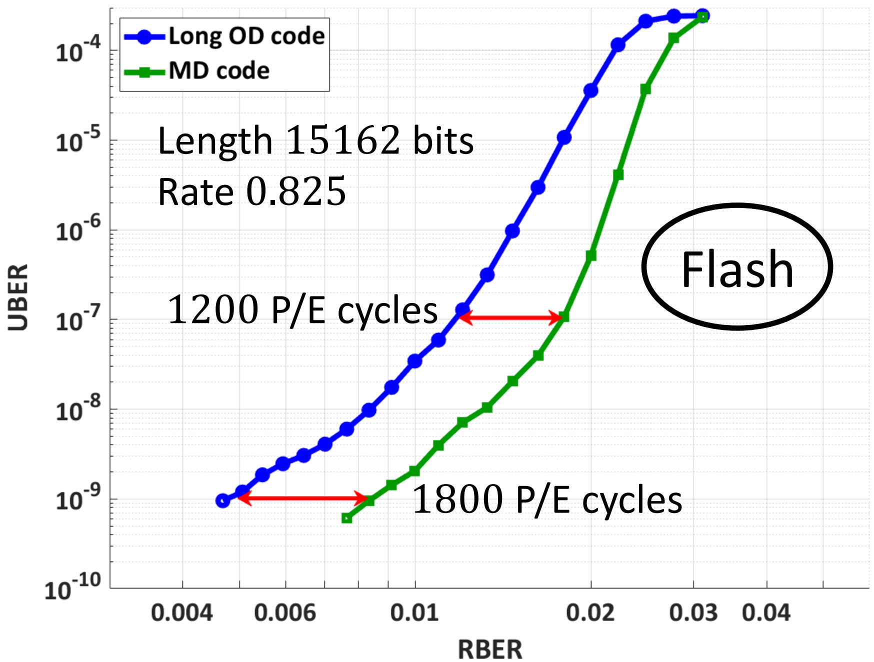


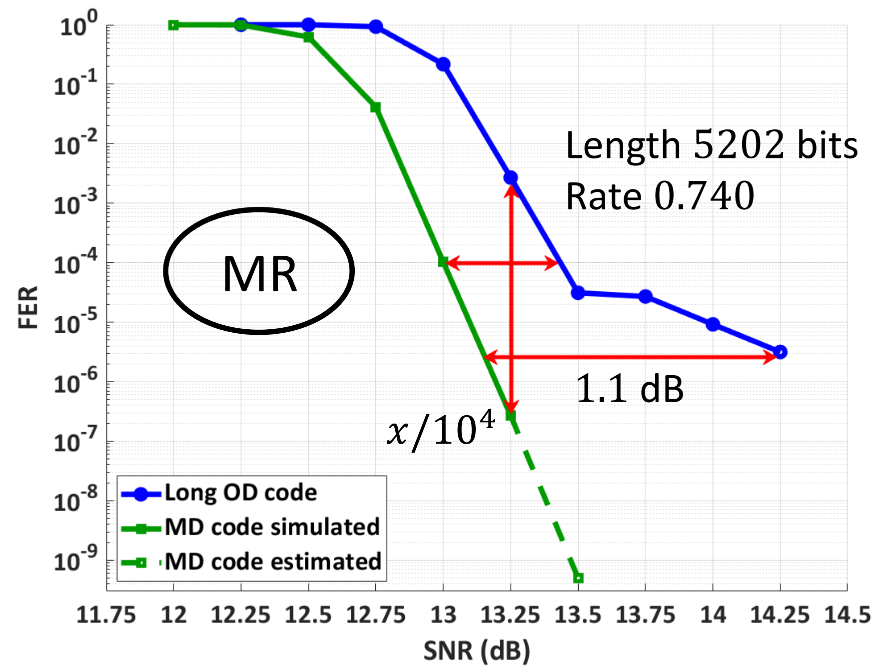




Significant Lifetime and Density Gains!

- **Effective MD coupling [H11]:**
  - Eliminates all instances of certain detrimental objects.
  - Achieves 1800 P/E cycles gain in Flash devices (left).
  - Achieves 1.1 dB and 4 orders of magnitude gain in MR devices (right).
  - These gains are vs. OD codes of the same parameters.





Observe threshold/waterfall gains: Opportunity in wireless.

My Related Work (1 of 2)

- [H5] **A. Hareedy**, C. Lanka, and L. Dolecek, “A general non-binary LDPC code optimization framework suitable for dense Flash memory and magnetic storage,” *IEEE J. Sel. Areas Commun.*, 2016.
- [H6] **A. Hareedy**, C. Lanka, N. Guo, and L. Dolecek, “A combinatorial methodology for optimizing non-binary graph-based codes: Theoretical analysis and applications in data storage,” *IEEE Trans. Inf. Theory*, 2019.
- [H7] H. Esfahanizadeh, **A. Hareedy**, R. Wu, R. Galbraith, and L. Dolecek, “Spatially-coupled codes for channels with SNR variation,” *IEEE Trans. Magn.*, 2018.
- [H8] H. Esfahanizadeh, **A. Hareedy**, and L. Dolecek, “Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting,” *IEEE Trans. Commun.*, 2019.
- [H9] **A. Hareedy**, R. Wu, and L. Dolecek, “A channel-aware combinatorial approach to design high performance spatially-coupled codes,” *IEEE Trans. Inf. Theory*, 2020.

My Related Work (2 of 2)

- [H10] S. Yang, **A. Hareedy**, S. Venkatasubramanian, R. Calderbank, and L. Dolecek, “GRADE-AO: Towards near-optimal spatially-coupled codes with high memories,” *ArXiv* and accepted at *IEEE ISIT*, 2021.

- [H11] **A. Hareedy**, R. Kuditipudi, and R. Calderbank, “Minimizing the number of detrimental objects in multi-dimensional graph-based codes,” *IEEE Trans. Commun.*, 2020.

- **A. Hareedy**, H. Esfahanizadeh, and L. Dolecek, “High performance non-binary spatially-coupled codes for Flash memories,” in *Proc. IEEE ITW*, 2017.

- **A. Hareedy**, H. Esfahanizadeh, A. Tan, and L. Dolecek, “Spatially-coupled code design for partial-response channels: Optimal object-minimization approach,” in *Proc. IEEE GLOBECOM*, 2018.

- **A. Hareedy**, R. Kuditipudi, and R. Calderbank, “Increasing the lifetime of Flash memories using multi-dimensional graph-based codes,” in *Proc. IEEE ITW*, 2019.

References (1 of 2)

- [R12] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.

- [R13] L. Dolecek, Z. Zhang, V. Anantharam, M. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, 2010.

- [R14] B. Amiri, J. Kliewer, and L. Dolecek, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," *IEEE Trans. Commun.*, 2014.

- [R15] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, 2013.

- [R16] D. G. M. Mitchell and E. Rosnes, "Edge spreading design of high rate array-based SC-LDPC codes," in *Proc. IEEE ISIT*, 2017.

References (2 of 2)

- [R17] M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce, and M. Baldi, “Design and analysis of time-invariant SC-LDPC convolutional codes with small constraint length,” *IEEE Trans. Commun.*, 2018.

- [R18] M. P. C. Fossorier, “Quasi-cyclic low-density parity-check codes from circulant permutation matrices,” *IEEE Trans. Inf. Theory*, 2004.

- [R19] T. Parnell, N. Papandreou, T. Mittelholzer, and H. Pozidis, “Modelling of the threshold voltage distributions of sub-20nm NAND flash memory,” in *Proc. IEEE GLOBECOM*, 2014.

- [R20] S. Srinivasa, Y. Chen, and S. Dahandeh, “A communication-theoretic framework for 2-DMR channel modeling: Performance evaluation of coding and signal processing methods,” *IEEE Trans. Magn.*, 2014.

Seminar Outline

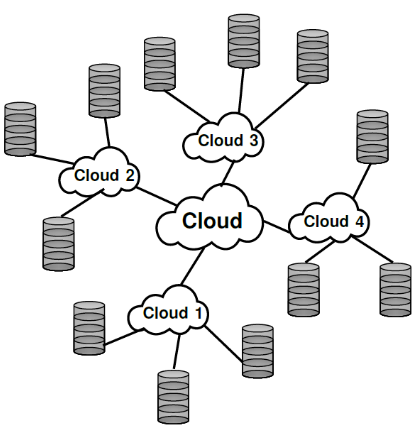
- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

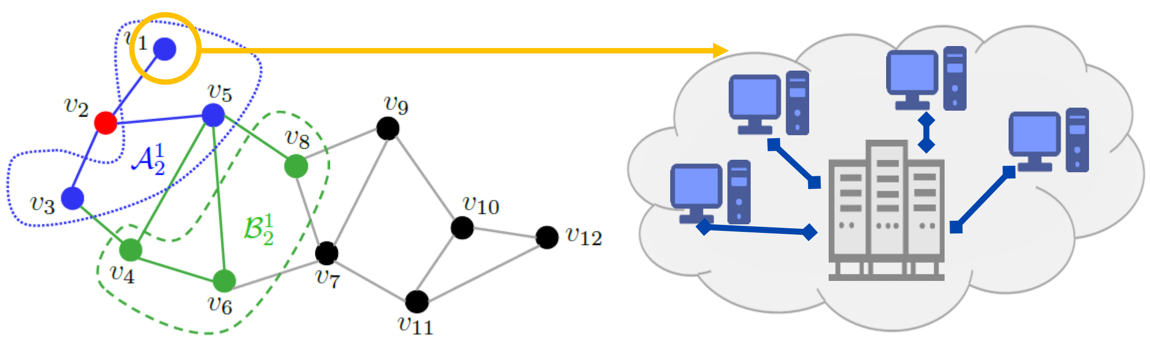
Types of Cloud Storage Systems

- **Centralized cloud storage:** A central cloud is connected to local clouds.
  - Only the central cloud owner can rent storage spaces to customers.
  - Examples: Amazon Web Services and Microsoft Azure.

- **Decentralized cloud storage:** No central cloud exists. No fixed topology.
  - Clouds can directly communicate, and users can rent storage spaces.
  - Examples: Blockchain-based cloud storage and Storj.

- **A codeword is distributed over multiple servers of the cloud.**
  - Failed servers (data erased) do not result in losing messages entirely.

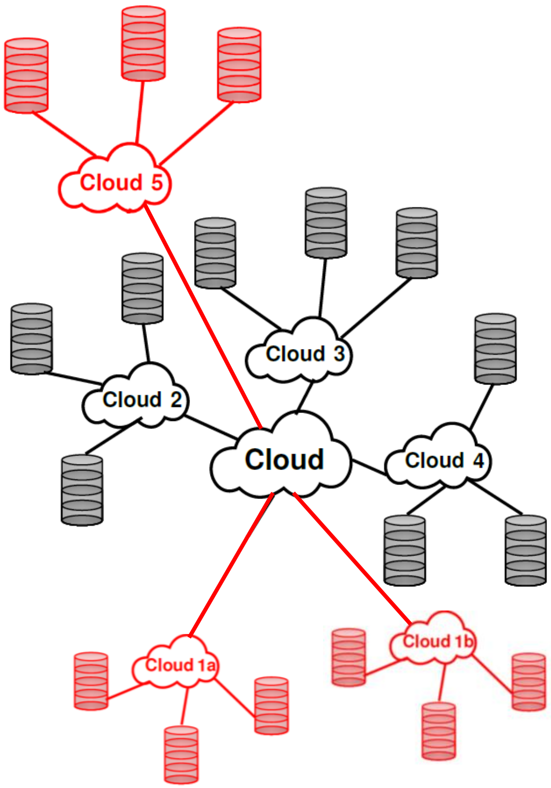




Supporting Scalability and Flexibility

- **Local and higher-level erasure-correction capabilities are provided.**
  - Higher-level capability via central cloud or via cooperation of clouds.

- **Our cloud storage solutions, which are based on algebraic coding, support:**
  - Scalability: New clouds are added with minimal changes needed to the existing system (cost saving).
  - Flexibility: A cloud that has its data suddenly becoming hot (of higher demand) can split into smaller, faster clouds.
  - Heterogeneity: Data lengths in various clouds are allowed to differ.
  - Topology-awareness: In the decentralized case, the solution adapts to the network topology.



My Related Work

- S. Yang, **A. Hareedy**, R. Calderbank, and L. Dolecek, "Hierarchical coding for cloud storage: Topology-adaptivity, scalability, and flexibility," *ArXiv* and submitted to *IEEE Trans. Inf. Theory*, 2020.

- S. Yang, **A. Hareedy**, R. Calderbank, and L. Dolecek, "Hierarchical hybrid error correction for time-sensitive devices at the edge," *ArXiv*, 2021.

- S. Yang, **A. Hareedy**, R. Calderbank, and L. Dolecek, "Hierarchical coding to enable scalability and flexibility in heterogeneous cloud storage," in *Proc. IEEE GLOBECOM*, 2019.

- S. Yang, **A. Hareedy**, R. Calderbank, and L. Dolecek, "Topology-aware cooperative data protection in blockchain-based decentralized storage networks," in *Proc. IEEE ISIT*, 2020.

Seminar Outline

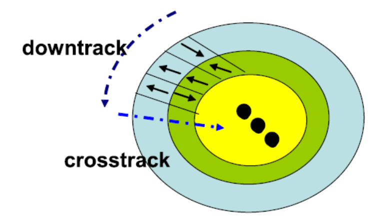
- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

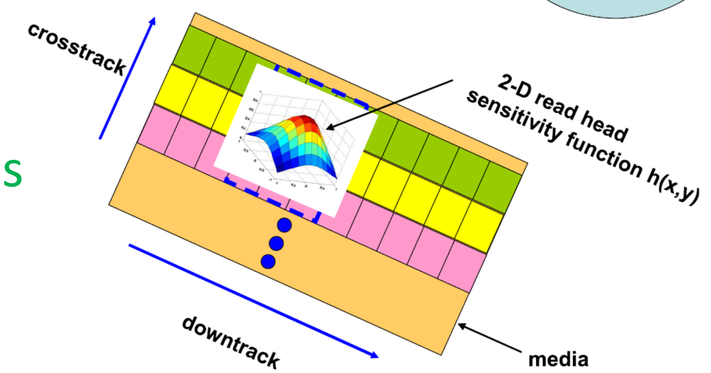
New Storage Channels Are Hard to Model

- **Ultra dense, next-gen storage devices have underlying channels with various effects to model.**
  - Examples on devices: V-NAND QLC/PLC Flash and TDMR devices.
  - Examples on effects: All effects contributing to MD non-uniformity.

- **Machine learning can help us break the barriers!**
  - The available mathematical models are quite complicated and do not capture everything.
  - Thus, coding solutions based on them can be notably improved.
  - I suggest using machine learning to direct the reconfiguration of LOCO codes and guide the design of LDPC codes.

TDMR [R20]





Machine Learning to Help Coding

- **Regarding constrained codes:**
  - As the device ages, error-prone patterns change.
  - We can learn the updated set of patterns to forbid from the LRs for errors collected at the output of the channel.
  - Next, we respond by reconfiguring the LOCO code (online).

- **Regarding error-correction codes:**
  - We can learn the set of detrimental objects from the LRs for errors collected at the output of the EC decoder.
  - Next, we design the LDPC code guided by that (offline).

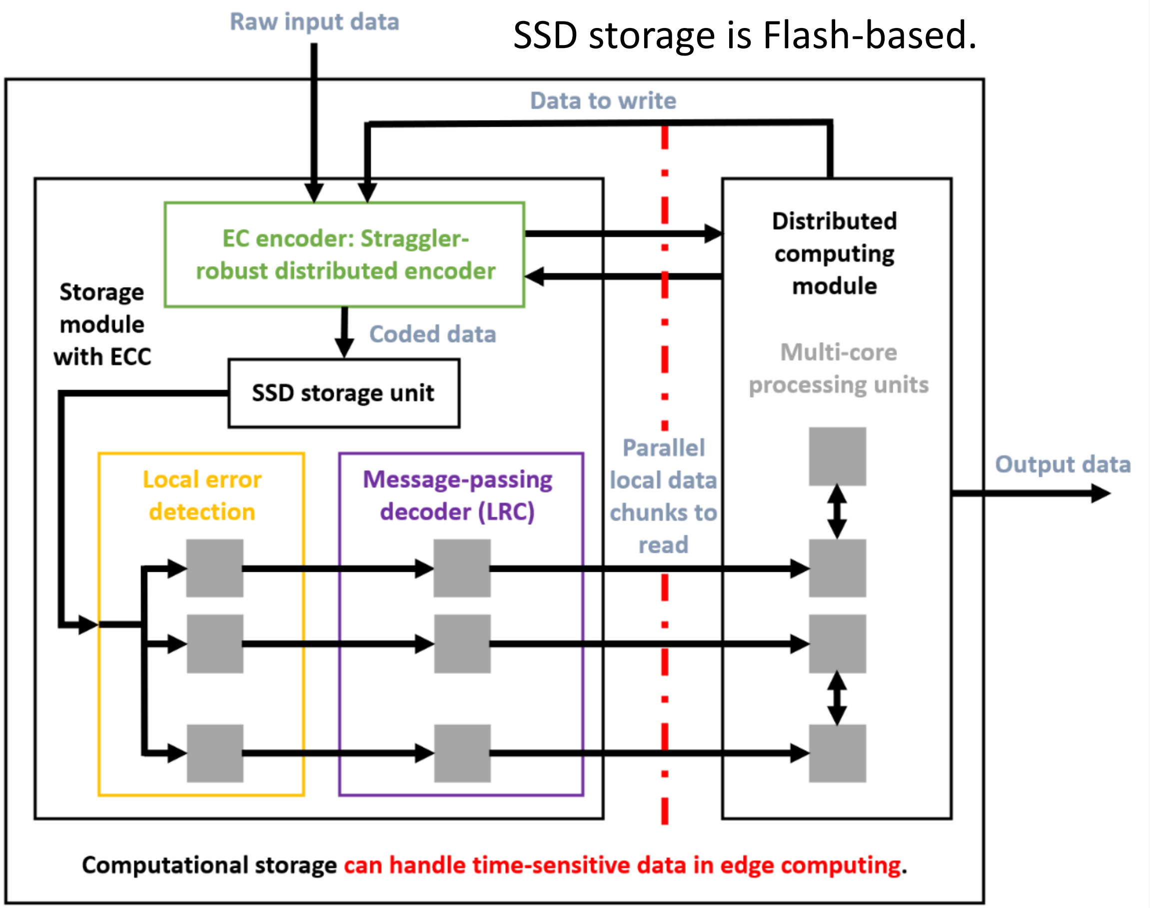
- **Significant lifetime gains can be achieved through these ideas.**
  - Machine learning can help improve detection and EC decoding as well.

- **This is a research direction I am following (with Duke and UCSD).**

A Framework for Computational Storage

- **Distributed machine learning promises lower latency, higher accuracy, and better scaling with large datasets.**
  - Computer architects have been searching for speed-up solutions, e.g., computing via GPUs.
  - One idea is to bring distributed computing units closer to data storage units.
- **I want to develop coding solutions that enable low-latency computational storage without compromising the reliability.**

SSD storage is Flash-based.



Coding to Help Machine Learning

- **Writing to the storage module:**
  - EC encoding can be performed distributively to speed up writing.
- **Reading from the storage module:**
  - Processing cores need not wait for an entire block to be decoded.
  - Message LRCs can significantly reduce the time to start computing.

Multi-level, adaptive EC capability

- **Speeding up distributed computing:**
  - If a worker straggles, the computation will not be completed.
  - Straggler-resilient coding handles this problem and reduces latency.

- **The above ideas can be applied via graph-based codes (high reliability).**

- **This is a research direction I am following (with Duke and UCLA).**

Seminar Outline

- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

Data Processing for DNA Storage

- **DNA storage can revolutionize data storage.**
  - Orders of magnitude gains in density and lifetime.

- **Stages of storing information are:**
  - DNA synthesis to generate the strands, storing these strands in a container, and sequencing to read.
  - All three stages suffer from errors.

- **External data processing includes:**
  - Clustering, sequence reconstruction, and error correction.

- **I want to develop novel data processing schemes for DNA data storage.**
  - Deep understanding of DNA characteristics is important.
  - Collaboration with other faculty members is crucial.

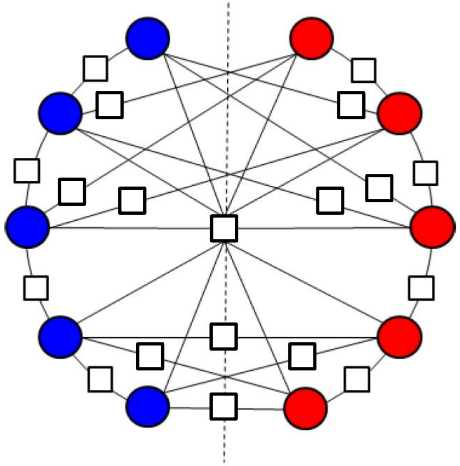
Coding for Quantum Systems

- **Quantum computers promise to solve problems remarkably faster than any classical computer.**
  - They are now becoming a reality.

IBM quantum computing system

- **Coding is required to ensure that computing and storage in quantum systems are performed reliably.**

- **I want to translate my classical results on high performance ECCs to the quantum world:**
  - Quantum LDPC codes are important.
  - Quantum absorbing sets degrade performance!



- **This is a direction I am following (with Duke and UA).**
  - Collaboration with other faculty members is crucial.

Seminar Outline

- Motivation and technical vision
- Reconfigurable constrained codes for data storage
- High performance graph-based codes
- Coding solutions for cloud storage
- How coding and machine learning can cooperate
- Challenges in DNA storage and quantum systems
- Conclusion and additional directions

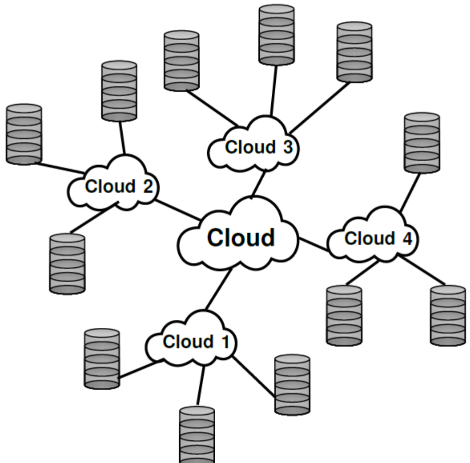
Takeaways and More Directions

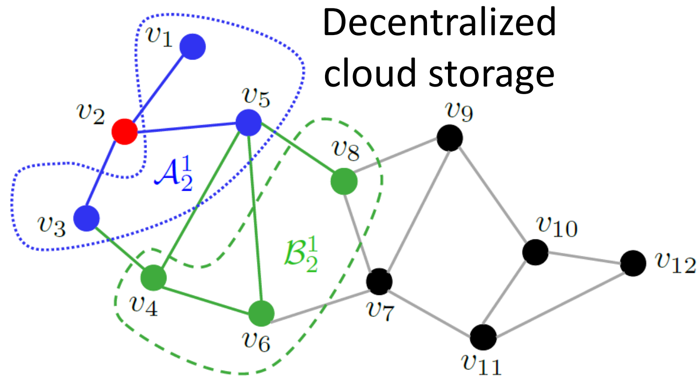
- **Conclusion:**
  - Storage densities are rapidly growing. Data require high protection.
  - LOCO codes exploit physics to fortify devices with minimal redundancy.
  - As the device ages, LOCO codes can be reconfigured to extend lifetime.
  - High performance SC codes are designed via OO/GRADE-AO techniques.
  - MD graph-based codes achieve significant lifetime and density gains.
  - Our coding solutions for cloud storage achieve scalability and flexibility.
  - Machine learning and coding can make the task of each other easier.
  - Advanced data processing improves DNA storage and quantum systems.

- **Additional research directions:**
  - MD-LOCO codes with MD-LDPC codes for MD storage devices.
  - Hierarchical algebraic codes for SSDs in multi-task systems.
  - Data processing methods for in-memory computing and analytics.

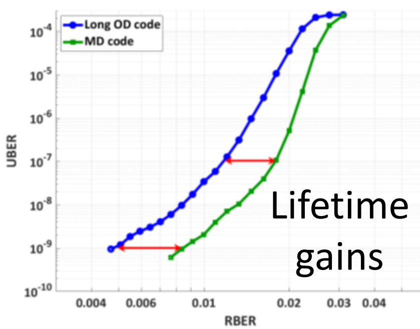
Today's Seminar in One Slide

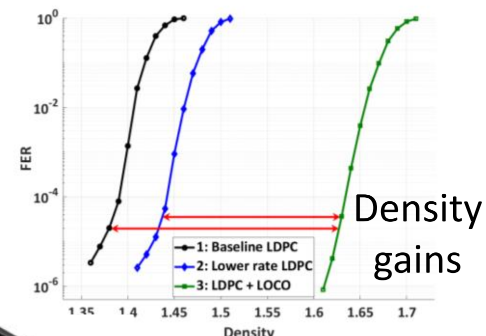
- **What did we talk about?**

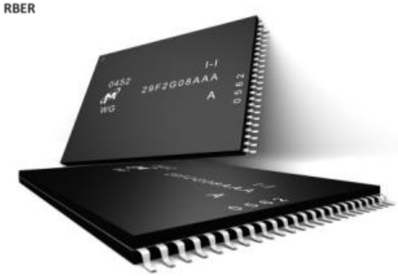




Machine learning helps coding







Coding helps machine learning

Thank You!