LDPC for SSD - From theory to practice

吴英全， Cody Wu

06/19/2022

Disclaimer

- This talk intends to provide engineering perspectives to bridge LDPC theory toward its system design in SSD controllers.
- The talk intentionally omits key technical details due to the intellectual property protection.

HDD vs SSD Controller Technology

	Hard-Disk-Drive	Solid-State-Drive
Back-end	Analog with memory	Memoryless binary (min data transfer)
DSP	ADC, Timing acquisition and sync, DFE, PRML(FIR, SOVA), Data-dependent Trellis branching, noise cancellation, and min-BER post-processing, etc.	Optimal threshold detection using Gaussian approximation
ECC	RS → Nonbinary LDPC	BCH → Binary LDPC
Page Size	512B → 4KB	2x4KB → 2x8KB → 4x16KB → 4x32KB
Front-end	SATA (3Gb/s), SAS(12Gb/s)	SATA(6Gb/s) → PCIe4x4(64Gb/s) → PCIe4x8(128Gb/s)
Random IOPs	~100 → ~200	~50K → ~800K → ~1.5M
Design Priority	1.SNR, 2.Power, 3.Thrput	1.Thrput, 2.Power, 3.SNR

ECC typically accounts for at least 25% area of controllers

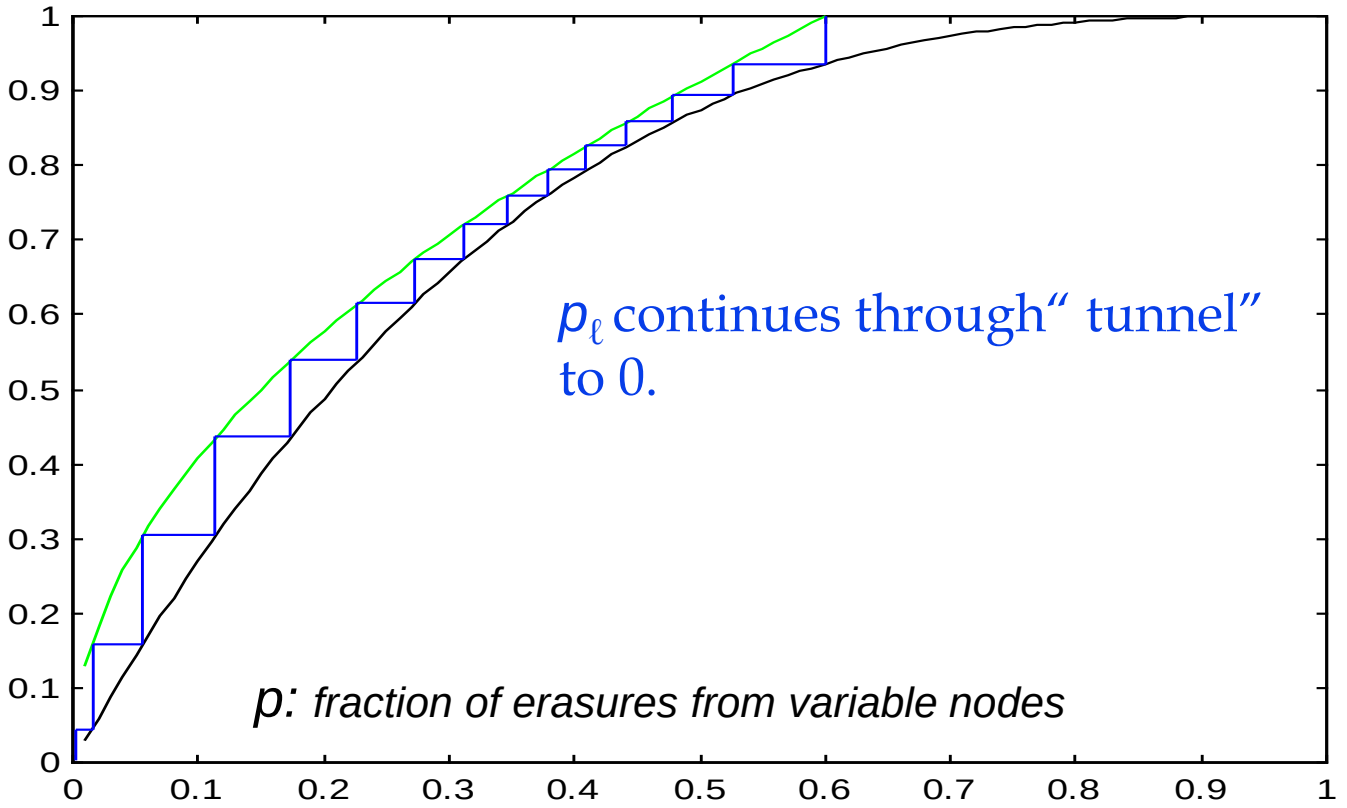
Finding Good Codes

- Shannon's capacity perspective:
  - Random code
  - Large block length
  - Optimal decoding
- Engineering perspective:
  - High throughput → structured code
  - Systolic design → structured encoder/decoder
  - System optimization → tradeoff among waterfall SNR, error floor, encoder/decoder complexity, throughput and power, etc.

LDPC Optimization by Density Evolution

- Example: (3,4)-regular LDPC code

q: fraction of erasures from check nodes

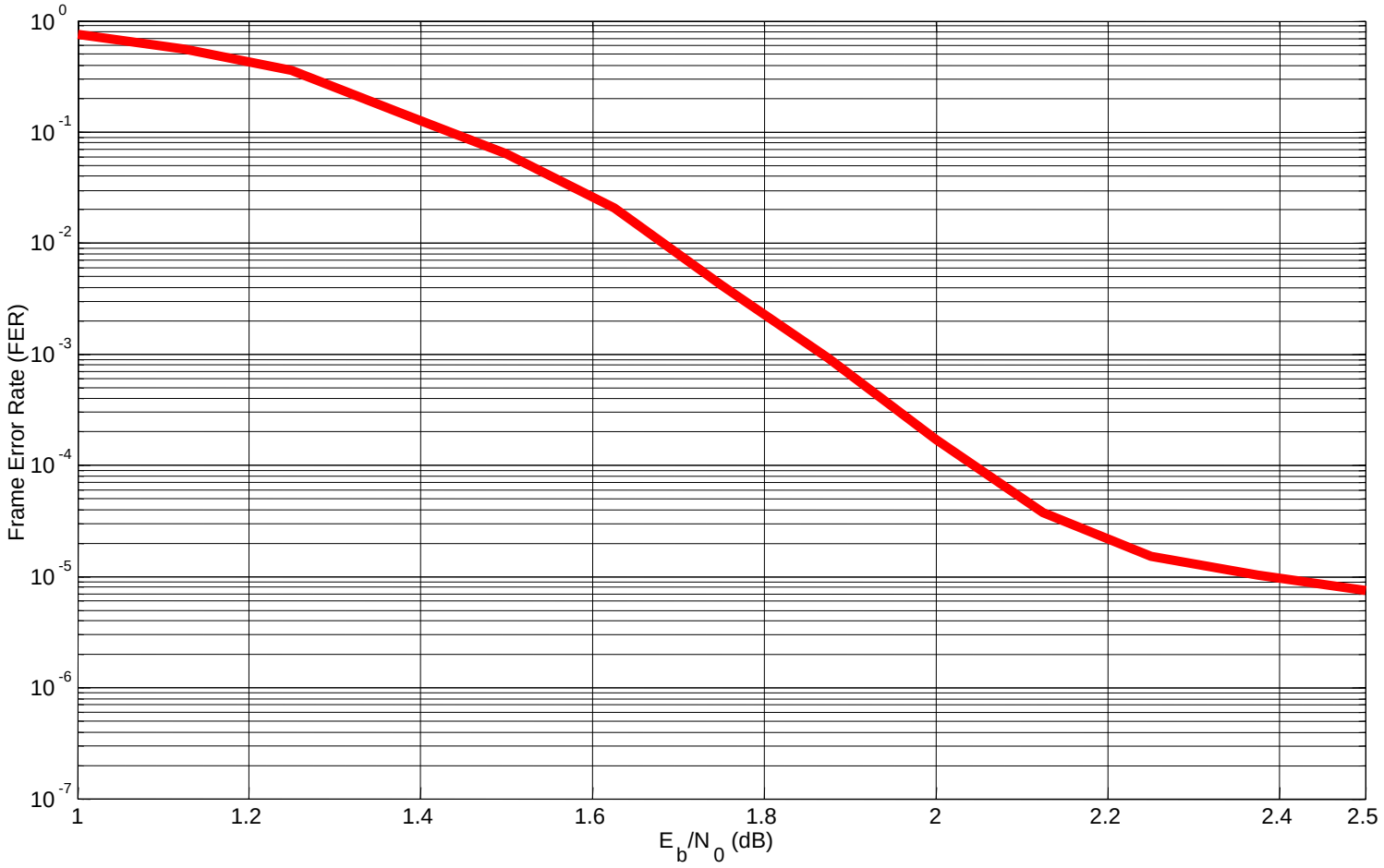


For a given rate, the objective is to optimize column and row degree distribution polynomials $\lambda(x)$ and $\rho(x)$ for the best threshold p^*.

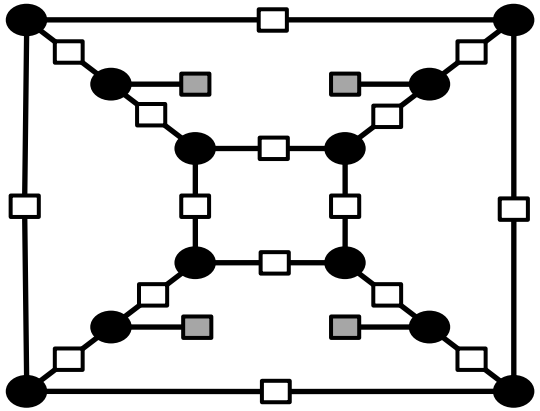
Limitation of Density Evolution

- Density Evolution assumes near infinite code length, and yields irregular degree distribution.
- Density Evolution optimizes waterfall, but not error floor. Low column degrees, particularly of degrees 1 and 2, are major components introduced by density evolution, while causing highly harmful trapping sets, accordingly, bad error floor.
- High column degrees, cause slow convergence and long latency. Moreover, practical (normalized) min-sum decoder is highly related to column weights, including the storage of V2C sign bits, width of adders, and width of variable total messages.
- Overall, density evolution offers little insight into LDPC design for data storage which requires very low error floor (below 1e-18 UBER).

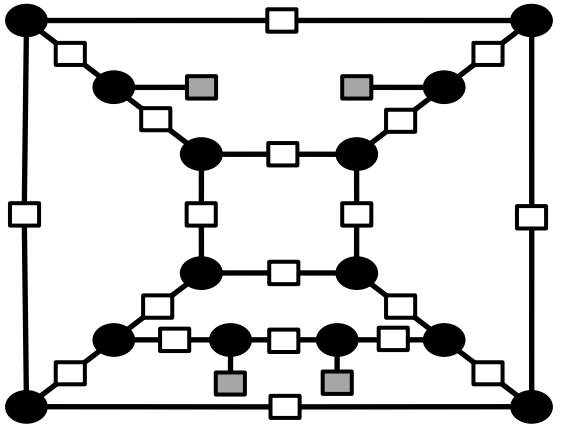
Error Floor and Trapping Sets



Rate-1/2, length-2640, Margulis code with SPA



(12,4) trapping set



(14,4) trapping set

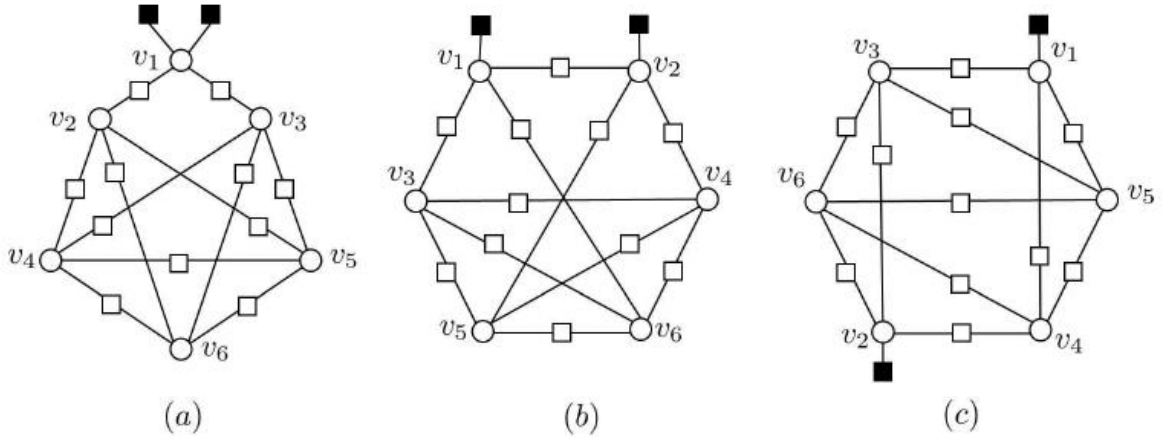


Fig. 2. All the possible non-isomorphic topologies for $(6, 2)$ ETSs in left-regular LDPC codes with $d_l = 4$ and $g = 6$.

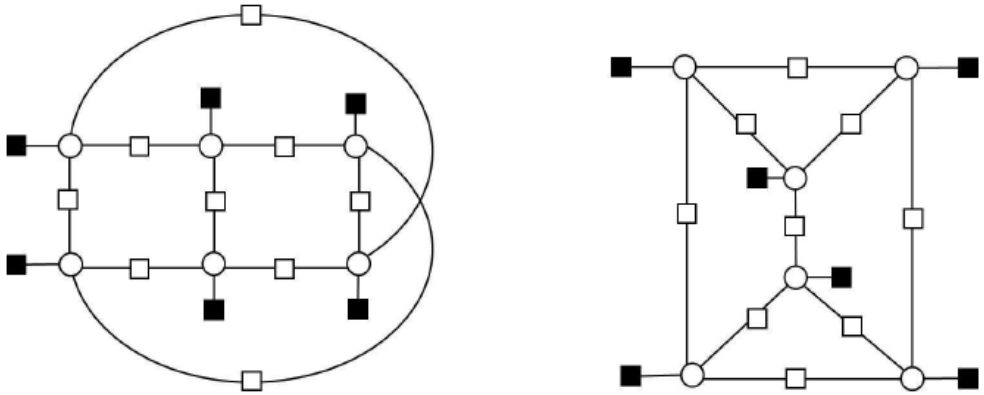


Fig. 6. Possible topologies for a $(6, 6)$ ETS in a left-regular graph with $d_l = 4$ and $g = 6$: the only two possible absorbing set topologies.

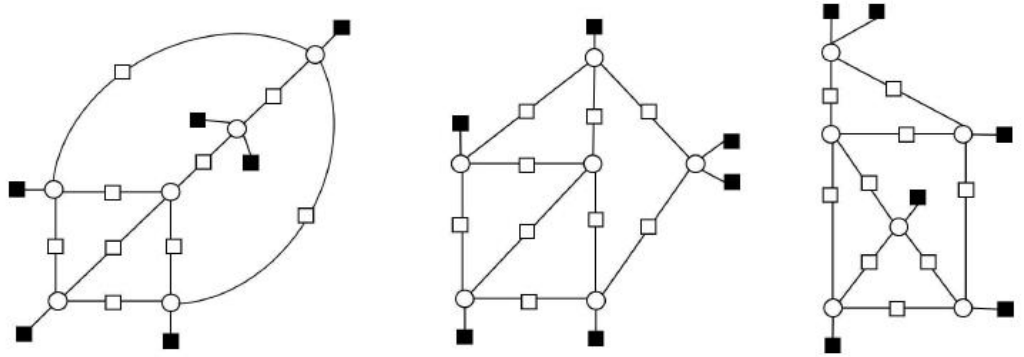


Fig. 7. Possible topologies for a $(6, 6)$ ETS in a left-regular graph with $d_l = 4$ and $g = 6$: topologies with only one variable node connected to two unsatisfied check nodes.

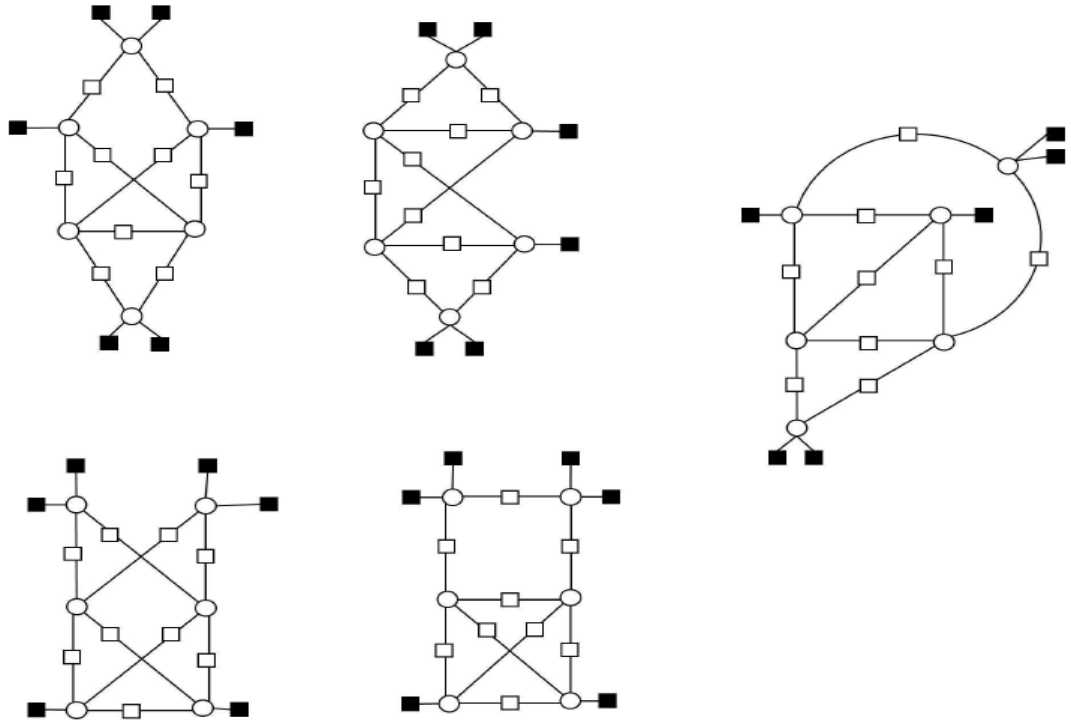


Fig. 8. Possible topologies for a $(6, 6)$ ETS in a left-regular graph with $d_l = 4$ and $g = 6$: topologies with only two variable nodes connected to two unsatisfied check nodes.

Challenges for Small Trapping Sets Elimination

- Trapping sets enumeration
  - Major research focuses on incremental expansion, say,

$$(3, 6)\rightarrow(4,6)\rightarrow(5,6)\rightarrow(6,6)\rightarrow(7,6)\rightarrow(8,6)$$
$$\downarrow\rightarrow(6,4)\rightarrow(7,4)\rightarrow(8,4)$$

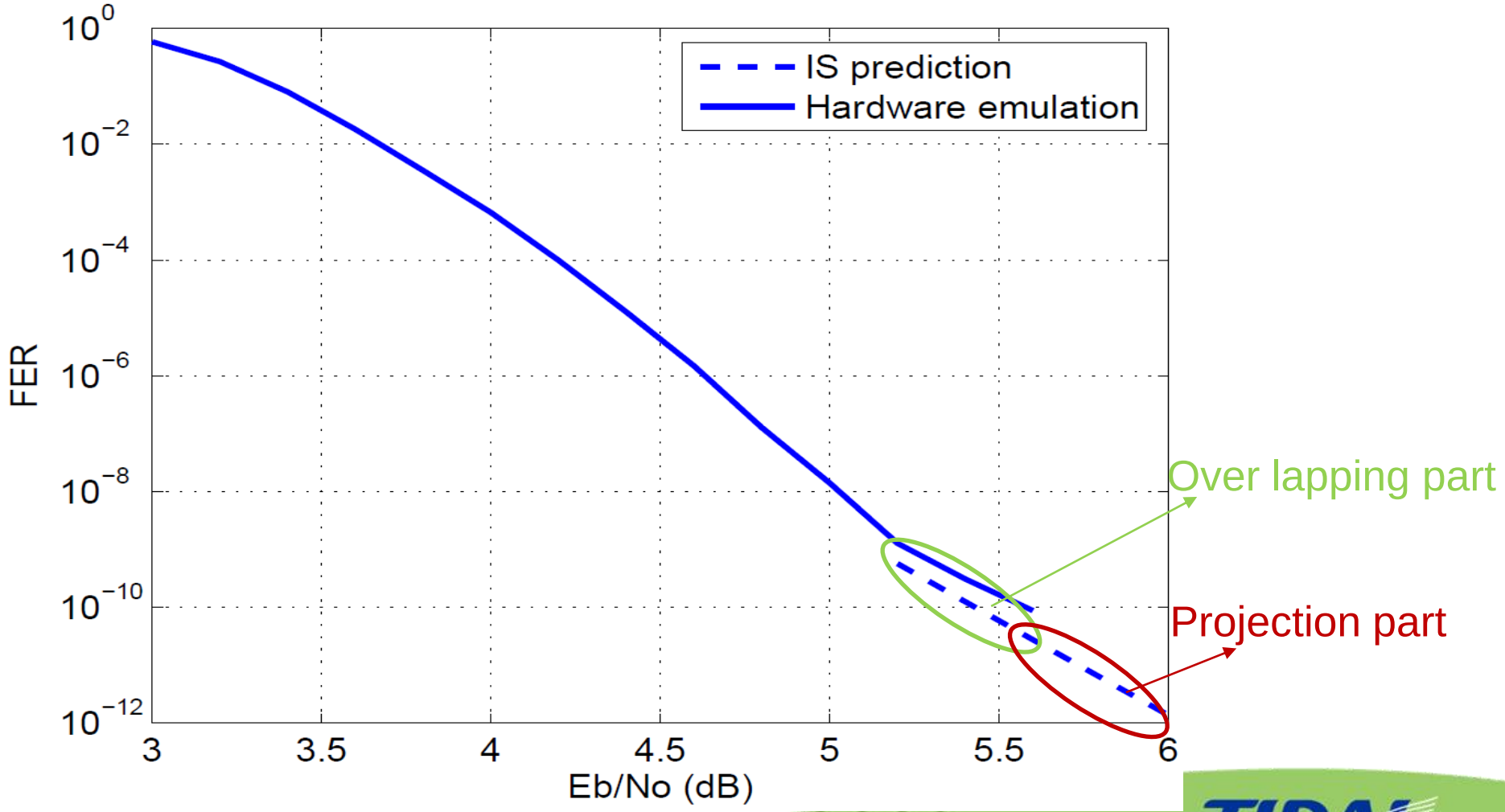
  - It must store all intermediate trapping sets, including unharmful ones. Storage is exploding for relative long codes, say 1KB. Moreover, large amount of time is consumed on eliminating repetitions.
- Trapping sets elimination
  - All researches are based on trial-and-error, i.e., randomly pick variable node edges, then search for resulting trapping sets. If bad ones are found, then trial again with random picks until successful or max trials. It is painful while no guarantee.
  - Researches have been focusing on cut-off optimization which is seriously flawed, i.e., by attempting to eliminate (a, b) sets, where $a \le A$, $b \le a$, whereas caring nothing on the cases $a>A$.

LDPC Error Floor Projection

- Using the mostly applied “Importance Sampling (IS)” technique to project the error correction performance of LDPC codes in low error rate region (BER below 10^{-12} and above which have exact Monte Carlo simulation results via software/hardware emulation)
  - Step 1: Get a list of the most harmful trapping sets
  - Step 2: Locating error boundary of the trapping sets
  - Step 3: Apply *importance sampling* to project low error rate performance

Example of Importance Sampling

- The overlapping area of MC simulation shall be consistent, and hence the projection is of higher reliability.



Two Significant Factors toward Practice

- Min-sum decoder
  - Avoid large table lookup and finite precision overflow
  - Greatly reduces storage (by storing only min1, min2, minIdx, VN sign)

- QC Structure
  - Greatly reduces ROM (for indexes)
  - Greatly reduces encoder hardware complexity
  - Greatly increases decoder parallelism and reduces circuitry

Existing algebraic LDPC construction has no added value

- A large branch of research has been on algebraic LDPC construction. To my view, existing methods do not offer any extra practical value, while imposing restrictions on code parameters.
  - There is no more efficient encoding or decoding method for any algebraic LDPC codes.
  - Existing algebraic structures are no more efficient than QC structure in implementation.
  - Code parameters are often against common practice, say, odd circulant size, inflexible information length.
  - The proven bounds on minimum distance and non-existence of small trapping set are way too loose to be meaningful.

Practical Dilemmas

- Low column degrees, particularly of degrees 1 and 2, cause highly harmful trapping sets, accordingly, high error floor.
- High column degrees, cause slow convergence and long latency. Moreover, practical min-sum decoder is highly correlated to column weights, including the storage of V2C sign bits, width of adders, and width of variable total messages.
- High-speed throughput calls for large circulant size, which only exists for long codes, whereas a long code requires linearly large decoder memory (memory typically accounts for 60-85% decoder area).
- Column-weight-4 H matrix is rank deficient, while existing efficient encoders utilizing H matrix entail full-rank H.
- Column-weight-5 code achieves about 0.1dB gain over the counterpart column-weight-4 code on hard-decoding, but looses 0.1 dB on soft-decoding. Moreover, the former entails an extra-bit VN-sign memory, extra and extra-bit adder, while converging 20% slower.

Missing puzzles from academics

- A simple but highly accurate metric to measure the performance of different LDPC codes (with same parameters), rather than monte-carlo simulations.
- A deep understanding of the tradeoff between encoder/decoder optimization and performance (waterfall and error floor) optimization. Randomness means inefficiency in hardware. QC-G matrix still yields a large circuitry for large codes, say, 1KB.
- A tight lower bound for minimum distance.
- More efficient construction beyond random search, say, progressive-edge-growth and various improvements, protograph lifting.
- Explicit (possibly algebraic) method to eliminate small trapping sets, rather than trial-and-error approach.

Bit-Flipping Preprocessing Is Meritless

- Bit-flipping decoder utilizes a circuit area of typically 10-15% of min-sum decoder.
- Bit-flipping decoder is about 1dB inferior to the normalized min-sum decoder, moreover, takes about 8-10x more iterations to converge.
- In my view, it is meritless to deploy bit-flipping decoder for early-life (fresh SSD). This is because,
  - Besides the above reasons, bit-flipping decoder can not replace, but rather supplement, min-sum decoder.
  - Our proprietary Beyond-Min-Sum™ decoder enables to converge in (roughly) 0.4 iterations in early life, effectively eliminating any power saving benefit provided by bit-flipping decoders.

Soft decoding benefit is tempting but hard to swallow

- 1-hard-bit-1-soft-bit decoding offers about 1.2dB gain over hard decoding in AWGN channel. 1-hard-bit-3-soft-bit decoding achieves about 1.5dB gain.
- Marketing/emulated soft-decoding assumes perfect soft information. However, when soft-reliability is distorted, the gain quickly diminishes, even worse, the mis-correction rate becomes intolerable (we observed 1e-8).
- It is foremost to ensure the hard read thresholds are optimal to provoke soft-read and soft decoding
- Due to low precision, say, 4-bit LLR, the soft-decoding optimized for waterfall is prune to suffer message-saturation, accordingly high error-floor, particularly due to defective cell bits.

Our Proprietary LDPC Encoder

- Regular column-weight-4 H matrix achieves good balance between low error floor, fast convergence, and low gate count.
- Column-weight-4 H matrix is rank deficient, while existing efficient encoders utilizing H matrix entail full-rank H.
- Conventional encoding using dense QC-G matrix suffers from large ROM (particularly to support multi-rate codes), large circuitry, and low throughput (typically 200-400 MB/s).

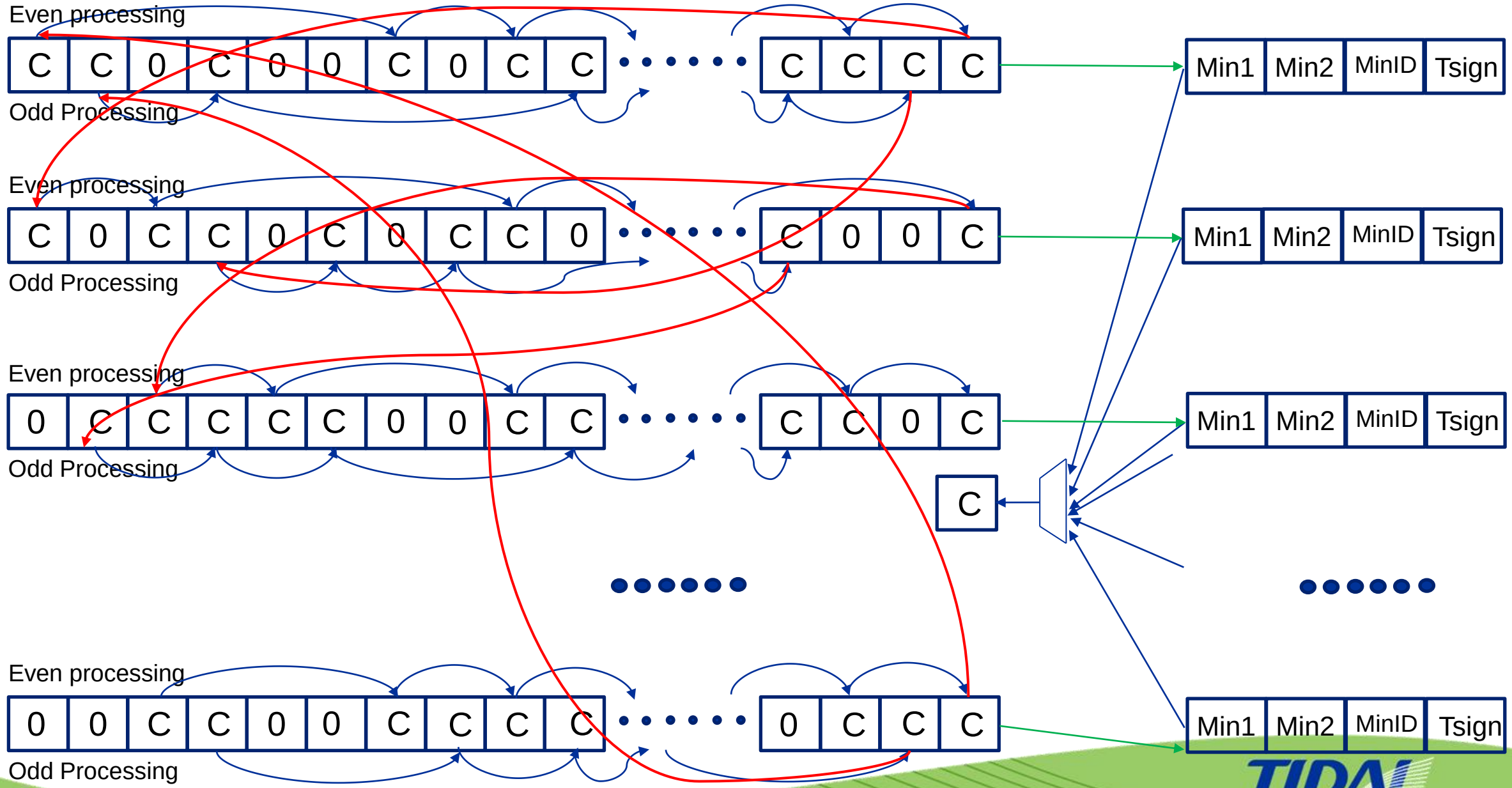
Code	Rate	G – dense circulants	(virtual) H^{-1} – dense circulants	Our dense circulants
(69, 64)x512	0.928	5x64= 320	5x5=25	≤6
(70, 64)x512	0.914	6x64= 384	6x6=36	≤6
(71, 64)x512	0.901	7x64= 448	7x7=49	≤6
(72, 64)x512	0.889	8x64= 512	8x8=64	≤6
(73, 64)x512	0.877	9x64= 576	9x9=81	≤6
(74, 64)x512	0.865	10x64= 640	10x10=100	≤6

Our novel encoder is built on new math foundation. It utilizes a highly sparse encoder matrix, yielding a gate count of 200K and a throughput of 16GB/s. The method can be extended to encode capacity-approaching LDPC codes with much lower complexity than the Richardson-Urbanke method.

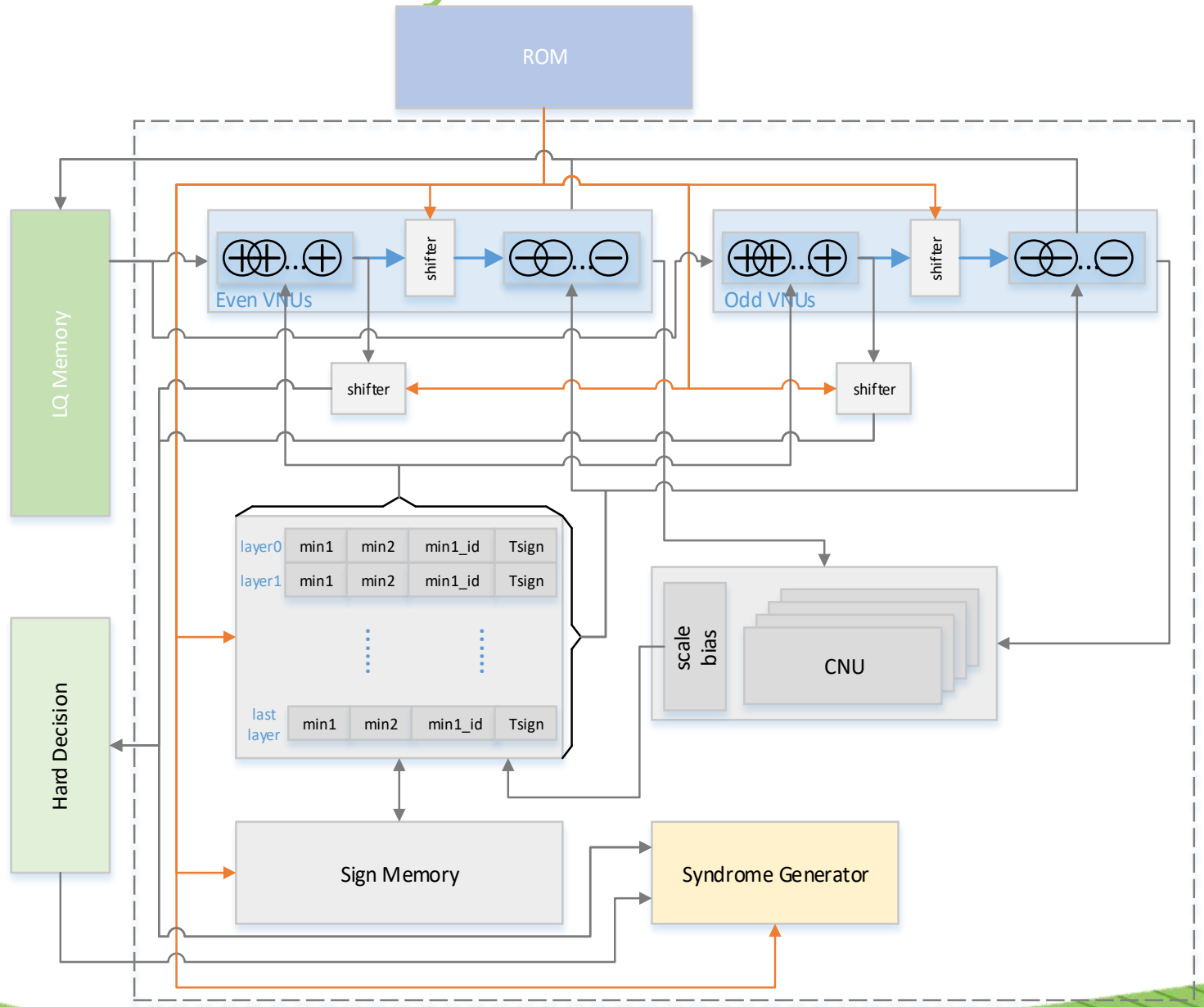
Our Proprietary LDPC Optimization

- We developed a new graph theory framework to construct LDPC codes algorithmically but deterministically, as opposed to the public-domain randomized searching.
- We discovered a new metric which leads to faster convergence and dramatically lower error floor. Our LDPC codes are mathematically proven to nearly optimize this new metric.
- We eliminate small harmful trapping sets, while minimizing the number of relatively large harmful trapping sets
- We ensure minimum distance is large enough (We constructed (69, 64)x128 code with dmin=41).
- We ensure to optimize the decoder area and throughput, by maximizing circuit utilization and reducing memory.
- Our specially designed rateless codes seamlessly fit into the single encoder and decoder.

Parallel Two-Circulant Layered Processing



Parallel Layered Decoder Architecture



Layered min-sum decoder SRAM usage (4-bit LLR)

SRAM	Standard-width	Ours
Hard-read	0.5x2 (ping-pang)	
Hard-dec	1x2 (ping-pang)	
Q-msg	2x6 (ping-pang)	
VN-sign	2x4 (2R +1W)	
Total	23 x Codeword	12 x

Our proprietary *Beyond-Min-Sum*™ decoder achieves 4-10x lower SFR, while lowering orders of magnitudes of error floor, over the normalized min-sum decoder

Layered Decoder Throughput

$$T_{thpt} = \frac{N_{byte} F_{clk}}{W_{col} N_{itr}} \square \frac{C_{data}}{C_{cw} + C_{idle}}$$

$L_{circ} = 256/8 = 32$ //number of bytes of a circulant

$N_{byte} = 2L_{circ} = 64$ //number of bytes processed per clock, 2 circulants in parallel

$F_{clk} = 1GHz$ //clock frequency

$W_{col} = 4$ //column weight in H matrix

$N_{itr} = 3$ //number of iterations

$C_{data} = 2K \square W_{col} / L_{circ} / 2 = 64$ //number cycles to process data circulants per iteration

$C_{cw} = C_{data} + N_{layer} W_{col} / 2$ //number of cycles to process codeword circulants

N_{layer} //number of H circulant layers

$C_{idle} = 10$ //number of idle cycles per iteration

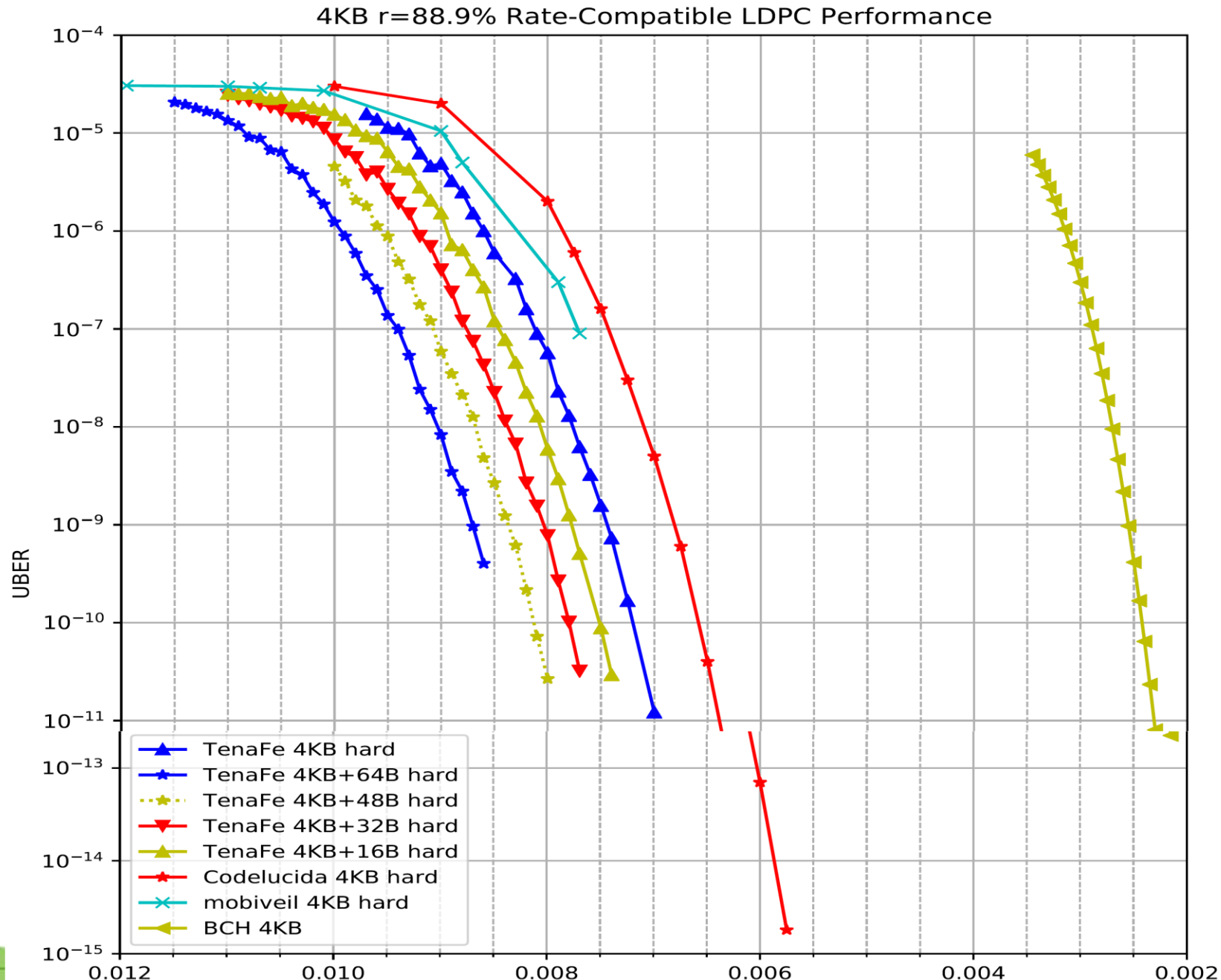
Code $(68,64)\times 128$: $N_{circ} = 4\times 68,\quad N_{idle} = 28,\quad T_{thpt} = 1.365\ GB/s$

Code $(70,64)\times 128$: $N_{circ} = 4\times 70,\quad N_{idle} = 10,\quad T_{thpt} = 1.412\ GB/s$

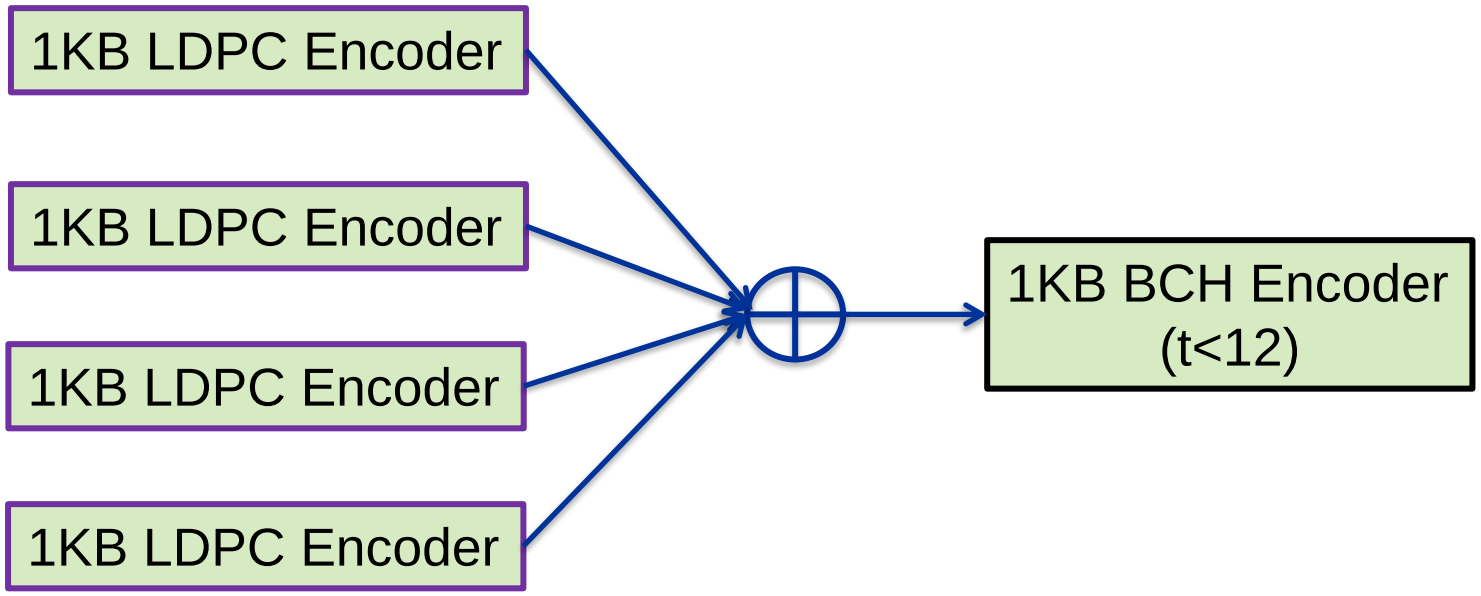
Code $(73,64)\times 128$: $N_{circ} = 4\times 73,\quad N_{idle} = 10,\quad T_{thpt} = 1.356\ GB/s$

Consider a column-wt-4 8-layer (72, 64) x128 code, wherein an average layer has 36 non-zero circulants, thus is processed in 18 cycles. On the other hand, high clock frequency of 1GHz is obtained through 7 pipelining stages which must take 6 idle cycles to flush out at the end of each layer, i.e. with 33% throughput drop-off. The key is on the code-decoder co-design to hide pipelining. We effectively achieved 0 idle cycle for entire iteration process

Rateless design to dynamically accommodate NAND overhead



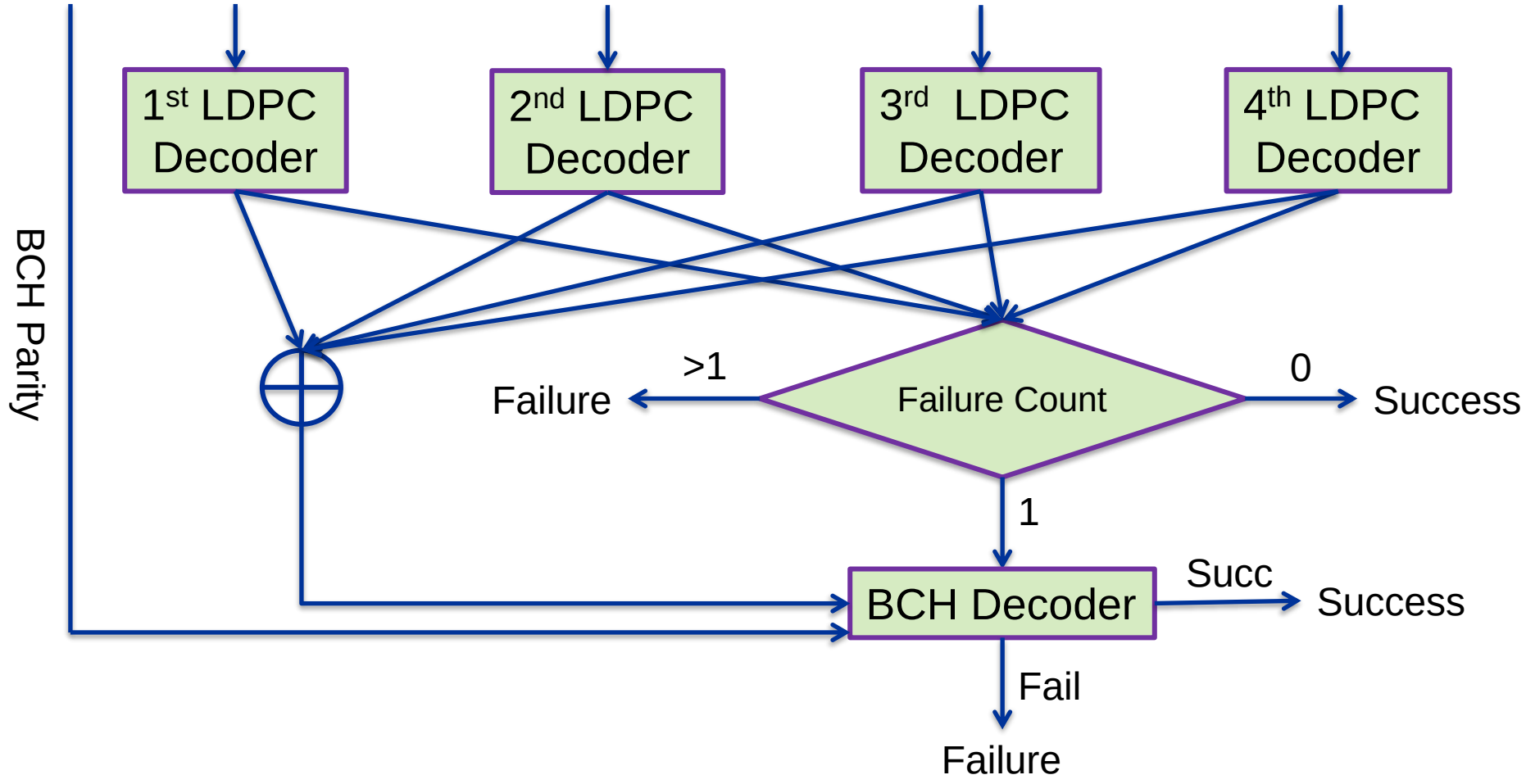
Simple Concatenation to Eliminate Error Floor



Quadratic error flooring

$$1 - 6 \times \left(1 - 10^{-10}\right)^2 \approx 10^{-19}$$

Simple Concatenation to Eliminate Error Floor



Conclusion

- Demonstrated engineering perspectives of LDPC design for SSD
  - High-throughput
  - Systolic design
  - System optimization across fields of waterfall SNR, error floor, high-throughput, low-power, etc.

Q & A