Flash Memory Summit

A Low-Cost LDPC Solution for Flash Memory

Osso Vahabzadeh, Jiangnan Xia, Paul Budnik

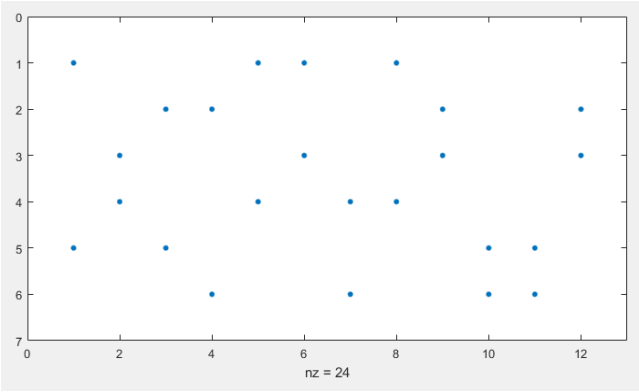
Symbyon Systems

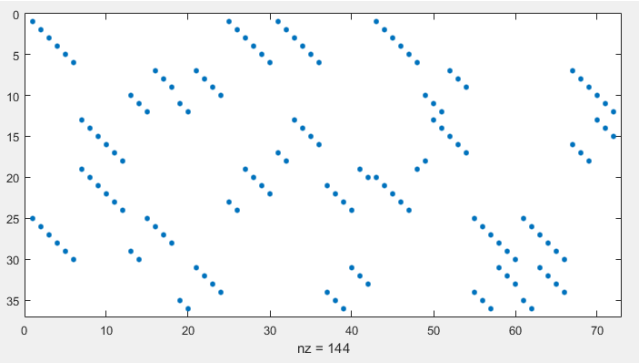
The material in this presentation is in part based upon work supported by the U.S. National Science Foundation under Award Number 1632562.

Outline

- Introduction
- Block Serial Layered Decoder Architecture and Requirements
- Simulation Results
- Conclusions

QC-LDPC Codes





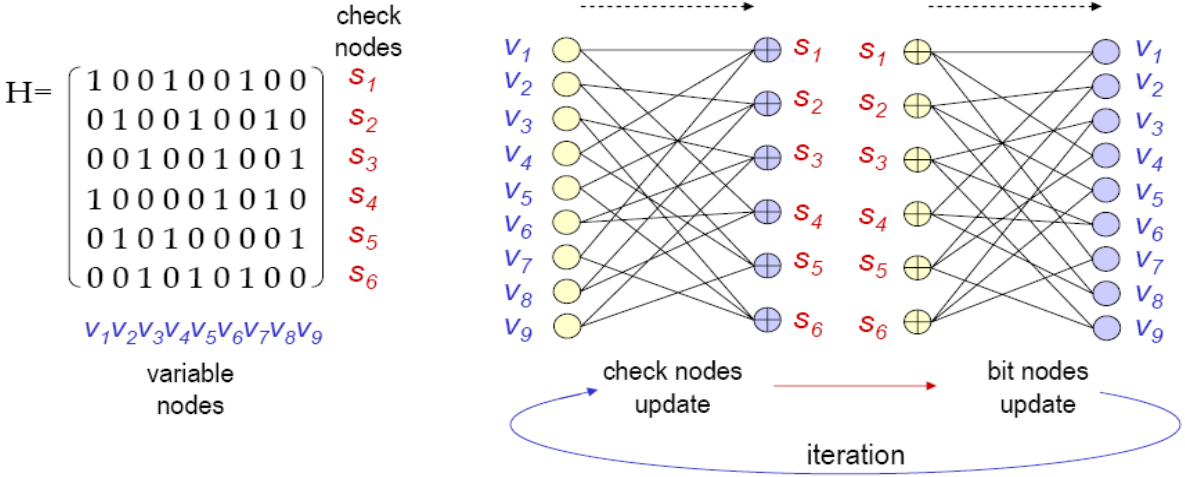

```
H_Base =

     1     0     0     0     1     1     0     1     0     0     0     0
     0     0     1     1     0     0     0     0     1     0     0     1
     0     1     0     0     0     1     0     0     1     0     0     1
     0     1     0     0     1     0     1     1     0     0     0     0
     1     0     1     0     0     0     0     0     0     1     1     0
     0     0     0     1     0     0     1     0     0     1     1     0
```

```
H_shift =

     0    -1    -1    -1     0     0    -1     0    -1    -1    -1    -1
    -1    -1     3     4    -1    -1    -1    -1     3    -1    -1     0
    -1     0    -1    -1    -1     4    -1    -1     5    -1    -1     3
    -1     0    -1    -1     4    -1     2     1    -1    -1    -1    -1
     0    -1     4    -1    -1    -1    -1    -1    -1     0     0    -1
    -1    -1    -1     4    -1    -1     3    -1    -1     3     4    -1
```

Iterative Decoding of LDPC Codes



- Variable/bit node reliabilities are initialization using the channel LLRs.
- Decoding performed iteratively where an iteration comprises check node and variable node updates followed by computing the bit node reliabilities and the hard decisions to estimate the codeword bits.
- Decoding stops when all parity equations are satisfied (success) or the maximum number of iterations is reached before the parity check condition is satisfied (failure).

Comparison of Decoder Architectures (1)

- Fully Parallel Architecture:
  - All check node messages are updated; then all variable node messages are updated; and so on.
  - All connections on the Tanner graph are physically implemented.
  - Huge hardware resources and routing congestion.
- Serial Architecture:
  - Check updates and variable updates done in a serial fashion using one unit only.
  - Minimal arithmetic and memory requirements.
  - Achieves very low throughputs.

Comparison of Decoder Architectures (2)

- Semi-Parallel Architectures:
  - Check node and variable node updates done using several units.
  - Partitioned memory by imposing structure on H matrix.
  - Practical solution for most applications.
  - Several semi-parallel architectures have been proposed.
  - Complexity differs based on architecture and scheduling.

Layered Decoder Architecture

- L and P messages are channel LLRs and overall reliability of the variable nodes, respectively,
- Q and R messages are variable and check node messages respectively.
- Optimized Layered Decoding with algorithm transformations for reduced memory and computations

$R_{l,n}{}^{(0)} = 0, P_n = L_n{}^{(0)}$ [Initialization for each new received data frame]

$\forall i = 1,2,\ldots, it_{max}$ [Iteration loop]

$\forall l = 1,2,\ldots, j$ [Block row or layer loop]

$\forall n = 1,2,\ldots, k$ [Block column loop]

$$R_{l,n}{}^{(i)} = f\left(\left[Q_{l,n'}{}^{(i)}\right]^{S(l,n')}\right), \forall n' = 1,2,\ldots, d_{c_l} - 1$$

$$(R_{\text{new}} = f(Q_{\text{new}}) = \text{R\_Select}(\text{FS}, \text{Qsign}))$$

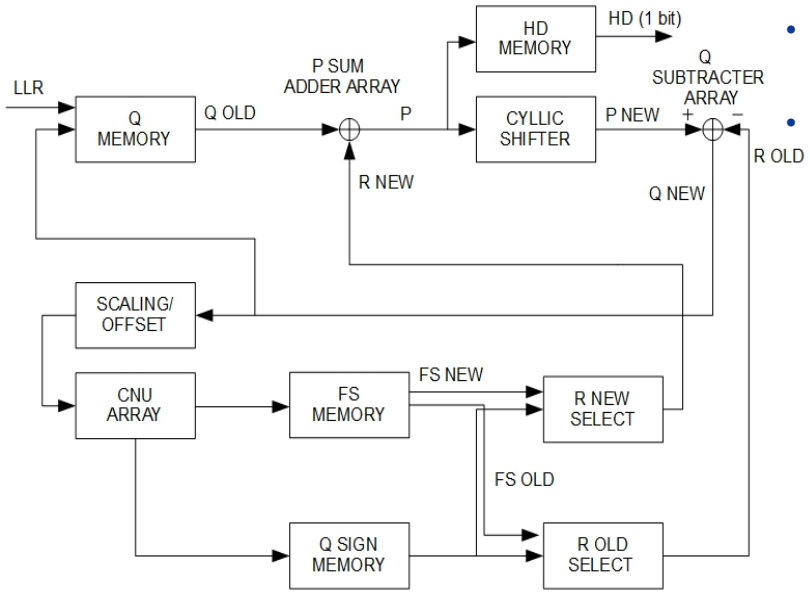
$$[P_n]^{S(l,n)} = \left[Q_{l,n}{}^{(i)}\right]^{S(l,n)} + R_{l,n}{}^{(i)}, \quad (P = Q_{\text{old}} + R_{\text{new}})$$

P_{new} is then computed by applying delta shift on P

$$\left[Q_{l,n}{}^{(i)}\right]^{S(l,n)} = [P_n]^{S(l,n)} - R_{l,n}{}^{(i-1)}, (Q_{\text{new}} = P_{\text{new}} - R_{\text{old}})$$

- $f(\cdot)$ is the check node processing unit
- $S(l, n')$ is the upward (right) shift for block row (layer) l and block column n'
- d_{c_l} is the degree of layer l

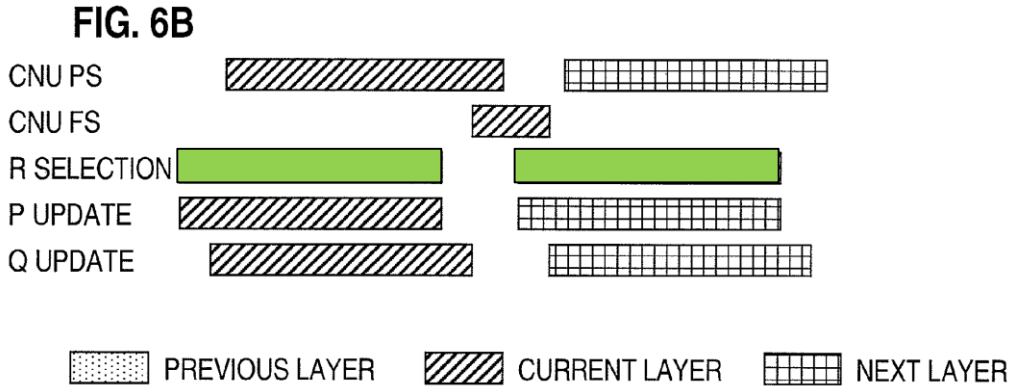
Block Serial Layered Decoder Architecture with On-the-Fly Computation



See [4, P1-P9] and references therein for more details on features and implementation.

- Proposed for irregular H matrices
- Goal: minimize memory and re-computations by employing just in-time scheduling
- Advantages compared to other architectures:
  1) Q (or L/P/Q) memory can be used to store L/Q/P instead of 3 separate memories- memory is managed at circulant level as at any time for a given circulant we need only L or Q or P.
  2) Only one shifter.
  3) Value-reuse is effectively used for both Rnew and Rold
  4) Low complexity data path design-with no redundant data path operations.
  5) Low complexity CNU design.
  6) Out-of-order processing at both layer and circulant level for all the processing steps such as Rnew and PS processing to eliminate the pipeline and memory access stall cycles.

Data Flow Diagram



R Selection for R NEW operates out of order to feed the data for PS processing of the next layer.

Illustration for out-of-order processing

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

- Rate 2/3 code. 8 Layers, 24 block columns. dv, column weight varies from 2 to 6. dc, row weight is 10 for all the layers.
- Non-zero circulants are numbered from 1 to 80. No layer re-ordering in processing. Out-of-order processing for Rnew. Out-of-order processing for Partial state processing.
- **Illustration for 2nd iteration with focus on PS processing of 2nd layer.**
- Rold processing is based on the circulant order 11 16 17 18 20 12 13 14 15 19 and is indicated in green.
- Rnew is based on the circulant order 72 77 78 58 29 3 5 6 8 10 and is indicated in blue.
- Q memory, HD memory access addresses are based on the block column index to which the green circulants are connected to.
- Q sign memory access address is based on green circulant number.
- Superscript indicates the clock cycle number counted from 1 at the beginning of layer 2 processing.

Out-of-order block processing for Partial State

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

- Re-ordering of block processing . While processing layer 2, the blocks which depend on layer 1 will be processed last to allow for the pipeline latency.
- In the above example, the pipeline latency can be 5.
- The vector pipeline depth is 5. So no stall cycles are needed while processing the layer 2 due to the pipelining. In other implementations, the stall cycles are introduced – which will effectively reduce the throughput by a huge margin.
- The operations in one layer are sequenced such that the block that has dependent data available for the longest time is processed first.

Out-of-order layer processing for R Selection

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	1	2			3	4		5	6		7	8					9	10						
1			11^{1}		12^{6}			13^{7}	14^{8}			15^{9}	16^{2}		17^{3}	18^{4}		19^{10}	20^{5}					
2			21	22		23		24		25		26		27	28				29	30				
3			31	32		33	34		35		36				37	38				39	40			
4	41		42			43	44			45		46		47			48				49	50		
5			51		52	53			54		55		56		57	58						59	60	
6	61	62		63		64			65			66		67	68								69	70
7		71	72				73		74	75		76	77		78		79							80

- Normal practice is to compute Rnew messages for each layer after CNU PS processing.
- Here the execution of R new messages of each layer is decoupled from the execution of corresponding layer's CNU PS processing. Rather than simply generating Rnew messages per layer, they are computed on basis of circulant dependencies.
- R selection is out-of-order so that it can feed the data required for the PS processing of the second layer. For instance Rnew messages for circulant 29 which belong to layer 3 are not generated immediately after layer 3 CNU PS processing.
- Rather, Rnew for circulant 29 is computed when PS processing of circulant 20 is done as circulant 29 is a dependent circulant of circulant of 20.
- Similarly, Rnew for circulant 72 is computed when PS processing of circulant 11 is done as circulant 72 is a dependent circulant of circulant of 11.
- Here the instruction/computation is computed at precise moment when the result is needed!!!

Memory Requirements (1)

- Q memory:
  - Width: circulant size * 8 bits
  - Depth: number of block columns.
- HD memory:
  - Width: circulant size
  - Depth: number of block columns.
- Qsign memory
  - Width: Circulant size * 1 bits and
  - Depth: Number of non-zero circulants in H-matrix.

Memory Requirements (2)

- FS memory:
  - Width: circulant size * (15 bits (= 4 bits for Min1 + 4 bits for Min2 index + 1 bit + 6 bits for Min1 index).
- FS memory access is expensive and number of accesses can be reduced with scheduling.
- For the case of decoder for regular mother matrices (no 0 blocks and no OOP): FS access is needed one time for Rold for each layer and one time for R new for each layer.
- For the case of decoder for irregular mother matrices: FS access is needed one time for Rold for each layer and one time for R new for each non-zero circulant in each layer.

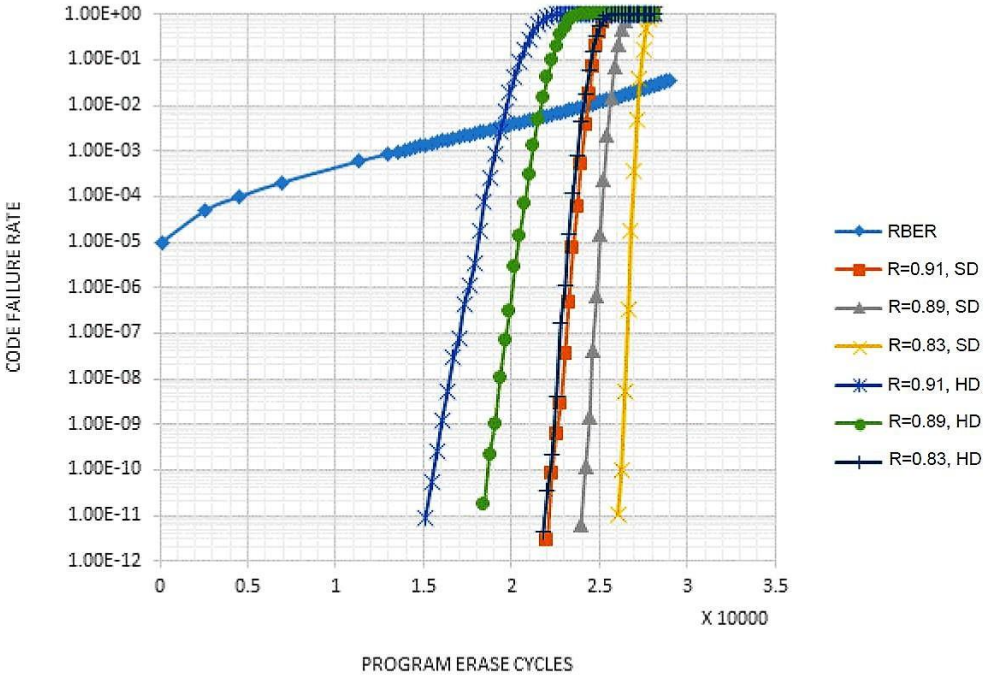
2 Circulant Processing

- Serves high-throughput applications while retaining the flexibility to support multiple codes
- Layered Update Module should support processing 2 circulants.
- The decoder design is similar to 1-circulant processing.
- Memory is organized in a slightly different way than 1-circulant processing.
- Q memory, HD memory, and Qsign memory:
  - Divided into 3 banks, the width remains the same, the depth is divided by 3

System Requirements and Design Specifications

- Requirements
  - Throughput in bits per sec.
  - BER
  - Latency
- BER would dictate Number of Iterations and degree profile (check node degrees and variable node degrees).
- Circulant Size (Sc)
- Number of Circulants processed in one clock (NSc)
- Throughput = Number of bits processed per clock × clock frequency
- Number of bits processed per clock = Sc × NSc × Rate / (Iterations × Average Variable Node degree)
  - Sc is usually set to less than 128 for smaller router.

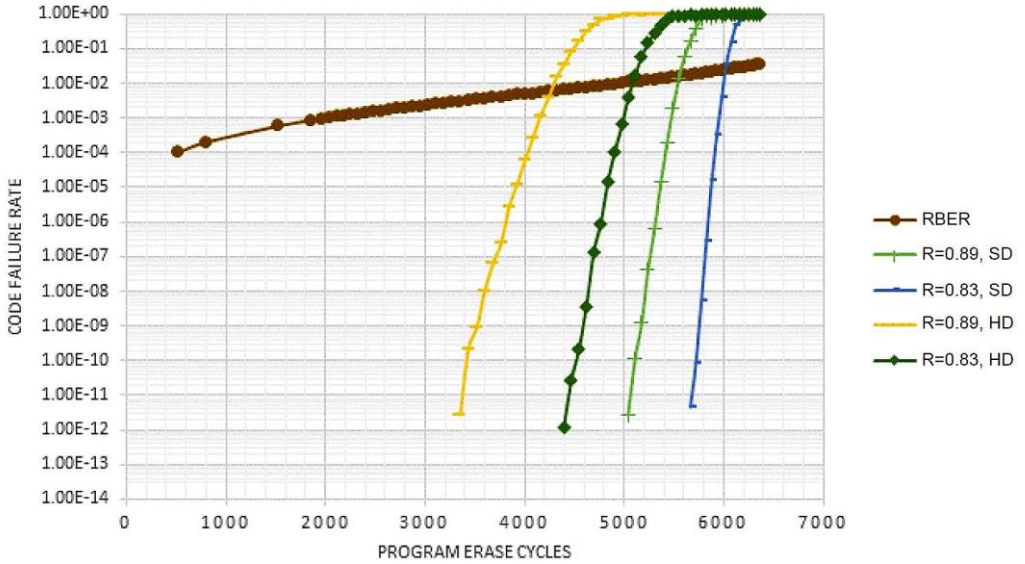
Flash Performance Results for Symbyon Layered Decoder (MLC 1Y Flash)



- Same 5-bit decoder is used in the HD and SD modes with different LLR tables.
- 5 strobe reads are used in the SD mode.
- Sets of packets are subjected to different P/E cycles and read test on each set is done after 90-day retention.
- 1KB codes with circulant size =140.

The results are courtesy of IEEE-SA P1890 Working Group on Error Correction Coding for Non-Volatile Memories

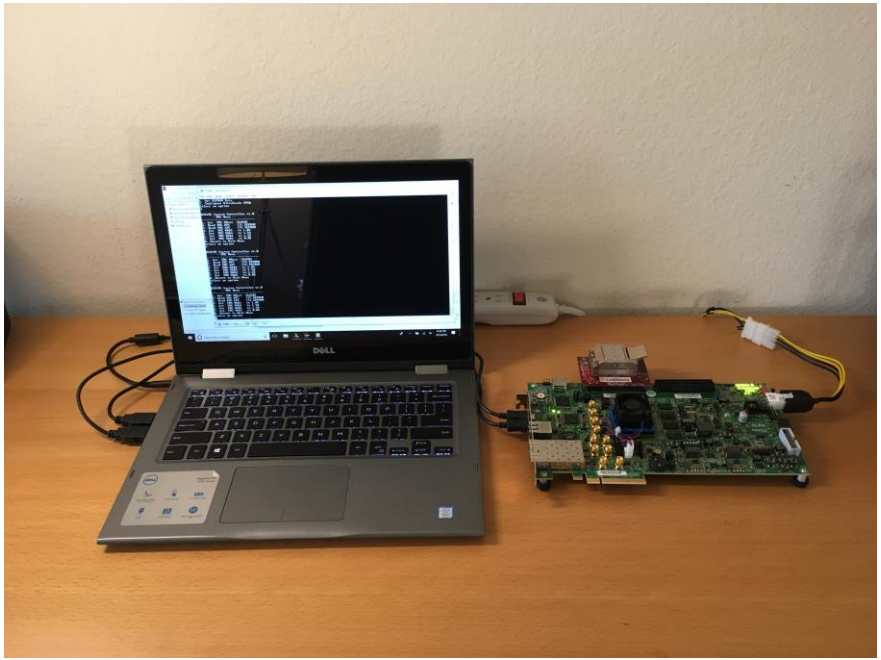
Flash Performance Results for Symbyon Layered Decoder (TLC 1Y Flash)



- 5 strobes generate 3 bits (1 HD bit + 2 SD bits).
- The system utilization can be reduced by going from 2 soft bits to 1 soft bit.
- However, data processing inequality states that no clever transformation of the channel output can give more information about the original data.
- Thus, 2 bits soft data is inherently more accurate than 1 bit soft data.

The results are courtesy of IEEE-SA P1890 Working Group on Error Correction Coding for Non-Volatile Memories

Simulation Setup for 3D Chips



The design has been successfully tested on 3D flash memories with Xilinx FPGA boards.

The system specifications depend on target throughput, latency, operating frequency, and operational error rate and P/E cycle range.

Conclusions

- LDPC decoder architectures offer different trade-offs in terms of throughput, area, memory requirements, and so on.
- Efficient decoder implementations require employing features such as datapath organization, out-of-order processing for Rnew, out-of-order processing for partial state, memory organization such as using the same memory for L, Q and P values, value-reuse property, check node unit designs, decoder scheduling features, code construction constraints for efficient hardware implementation and numerous other features, which are equally applicable to binary and non-binary decoders.
- The block serial layered decoder is an efficient decoder architecture that benefits from the above mentioned features and has been successfully tested with planer and 3D flash memories.

References [1]

1. Blanksby, A., and Howland, C., “A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder”, IEEE J. Solid-State Circuits, Vol.37, Iss.3, pp. 404-412, Mar 2002.

2. Levine, *et. al.,* “Implementation of near Shannon limit error-correcting codes using reconfigurable hardware,” *IEEE Field-Programmable Custom Computing Machine,* 2000.

3. E. Yeo, “VLSI architectures for iterative decoders in magnetic recording channels,” *IEEE Trans. Magnetics*, vol. 37, no.2, pp. 748-55, March 2001.

4. K. K. Gunnam, “LDPC Decoding: VLSI Architectures and Implementations,” *Flash Memory Summit,* 2013.

5. Proposal Draft on H Matrix and Decoder Choice for P1890 LDPC Decoders, IEEE P1890 Standards Working Group on Error Correction Coding for Non-Volatile Memories, July 2014, accessed May 4, 2018, https://ieee-sa.imeetcentral.com/p1890wgpublic/raw/H%20Matrix%20and%20Decoder%20Choice%20for%20IEEE%20Flash%20Standard.pptx

Some of the slides in this presentation are used from [3] with permission.

References [2]

Efficient decoder implementations require employing features such as datapath organization, out-of-order processing for Rnew, out-of-order processing, memory organization such as using the same memory for L and Q values and using the same memory for L and P values, value-reuse property, check node unit designs, decoder scheduling features, code construction constraints for efficient hardware implementation and numerous other features, which are equally applicable for binary and non-binary decoders. These features are covered by several issued and pending patents owned by Texas A&M University System (TAMUS) and, as observed by IEEE-SA P1890 Working Group, they provide memory and logic savings up to 75% compared to standard LDPC implementation (see [4]). These patents and the copyrighted software are controlled by TexasLDPC Inc. dba Symbyon Systems and can be jointly or separately licensed from them directly. There is a use case where the technology presented in this presentation is licensed for use in high volume hard disk drives (HDDs). The following is the list of the above referenced issued and pending patents:

[P1] US8418023B2: Low density parity check decoder for irregular LDPC codes

[P2] US8656250B2: Low density parity check decoder for regular LDPC codes

[P3] US8555140B2: Low density parity check decoder for irregular LDPC codes

[P4] US8359522B2: Low density parity check decoder for regular LDPC codes

[P5] US9112530B2: Low density parity check decoder

[P6] US10141950B2: Low density parity check decoder

[P7] US20170093429A1: Low density parity check decode

[P8] Provisional Application No. 60/988680

[P9] Provisional Application No. 60/915320

Thank You! Questions?