Improving Waterfall Performance of low-cost FAID LDPC Decoders

D. Declercq

Contact: declercq@codelucida.com

Website: www.codelucida.com

Outline

1. FAID decoding for LDPC codes

2. Improving Error Correction Performance with **more Iterations**

3. Improving Error Correction Performance with **more Precision**

Low Cost LDPC Decoder: FAID™

- Storage using 3D TLC / QLC Flash requires **LDPC ECC** for the improved performance against BCH
- LDPC is especially important to improve the **endurance** and the robustness to **retention**
- Strong Error Correction Coding (ECC) is needed to **limit the number of page reads** and **extend the life** of the Flash memory.
- The very strong ECC needs to come at **low Hardware Costs** and cope with the **increasing throughputs** of the fastest interfaces

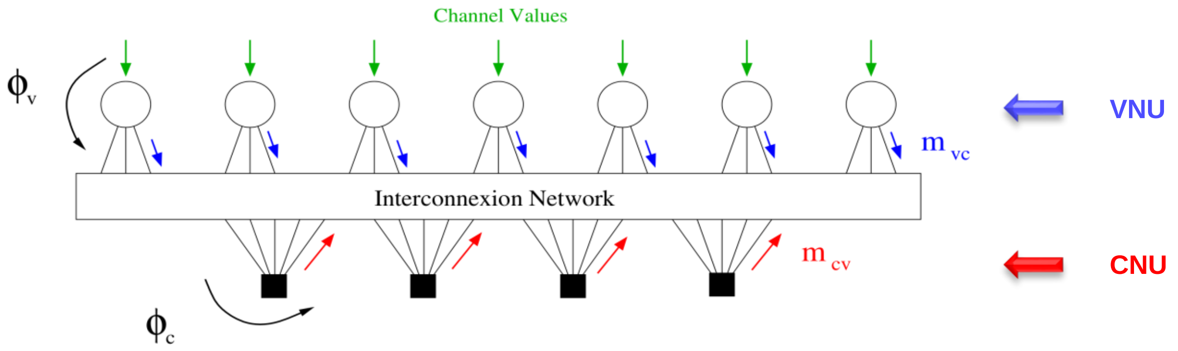
FAID™ : Finite Alphabet Iterative Decoding

Codelucida

FAID decoding

Originally Introduced using 3-bits message precision

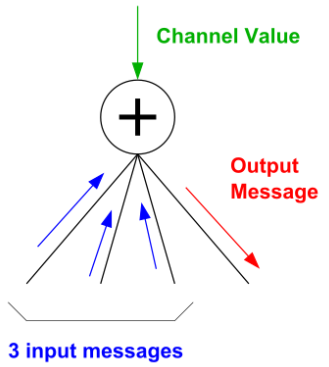
- **Regular Quasi-cyclic LDPC codes** - parity-check matrix defined by circulant blocks
- Iterative decoders with 3-bits messages belonging to $\mathcal{A}$ **= { -3 , -2 , -1 , 0 , +1 , +2 , +3 }**
- One iteration comprising **Variable-Node Updates (VNU)** and **Check-Node Updates (CNU)**
- **Vertical** column-wise layered **scheduling** for low memory and fast processing

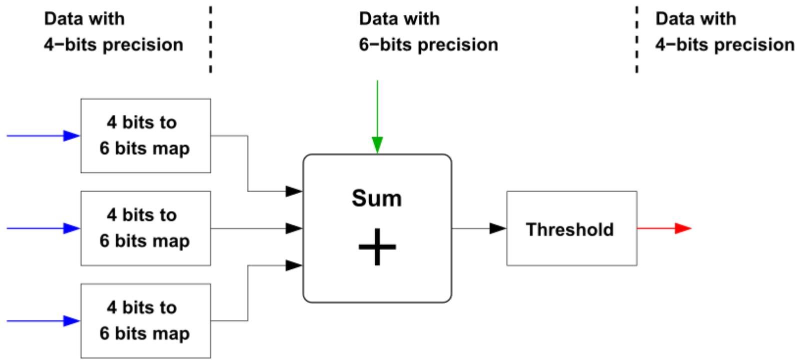


Codelucida

Classical Solution: Min-Sum Variable Node Implementation

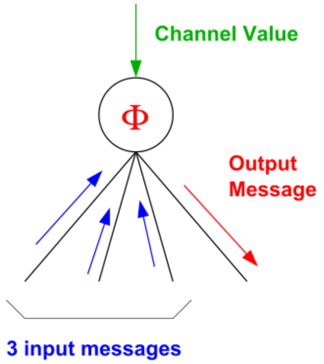
- For regular column weight $d_v=4$, the **VNU** takes **3 input messages** and together with the **channel value**, generates a **4th output message**,
- Messages for Min-Sum decoding use typically **4 precision bits**, and the sum uses **6 precision bits**
- Output of the VNU is a message with **4 precision bits**

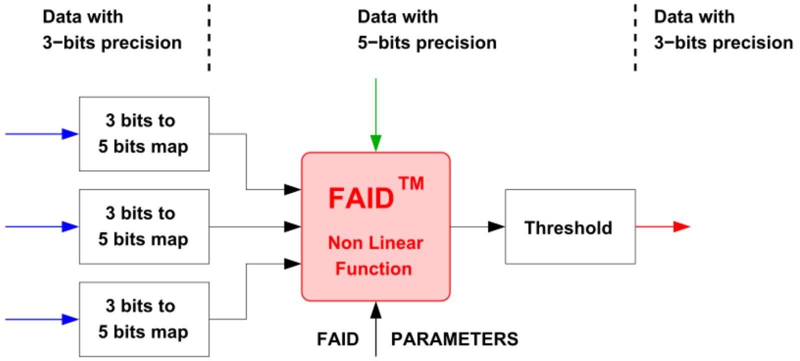



Data with 4–bits precision
Data with 6–bits precision
Data with 4–bits precision
4 bits to 6 bits map
4 bits to 6 bits map
4 bits to 6 bits map
Sum
Threshold

Our Solution: FAID Variable Node Implementation

- The SUM operator of the **VNU** is replaced by a carefully optimized non-linear Boolean function **Φ**
- Messages for FAID decoding use typically **3 precision bits**, and the non-linear function is defined with **5 precision bits**
- Output of the VNU is a message with **3 precision bits**



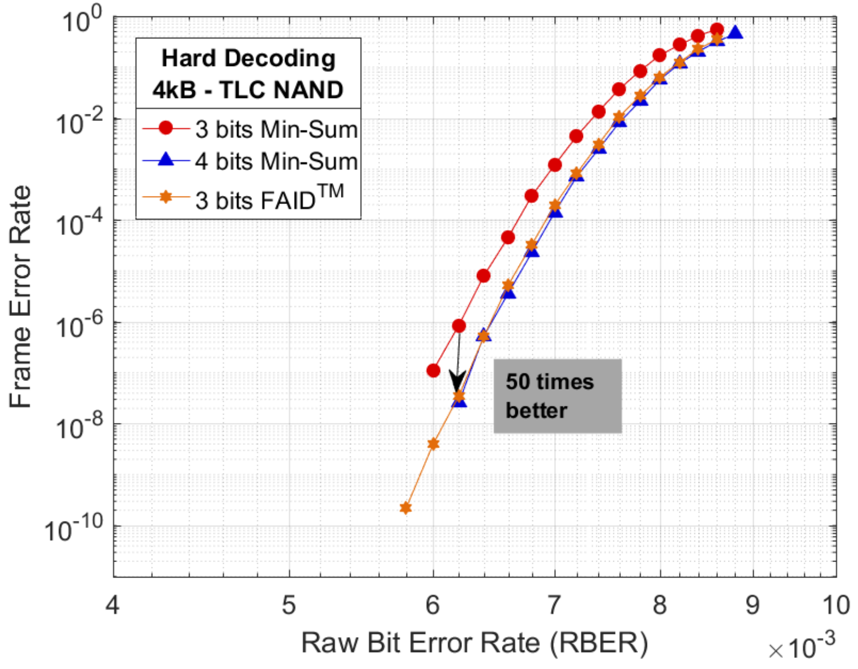


Why FAID have good ECC performance ?

- Low precision iterative decoders tend to **limit** the ECC performance, both in the waterfall and Error floor regions

- **FAID approach:** low precision = **3-bits**, but **optimize** the non-linear VNU function to recover the performance loss

- **FAID with 3-bits precision** has the same ECC performance as **4-bits Min-Sum**



Can we Improve the ECC performance ?

- FAID already uses optimized VNU functions for 3-bits precision

  ⇒ not much degree of freedom

- No ECC gain can come from different scheduling (vertical layered, horizontal layered, flooding)

- Improved ECC could come from **larger maximum number of iteration**

  ⇒ **Challenge:** tradeoff between worst case latency vs. ECC gain

- Improved ECC could come from **extra precision for the messages**

  ⇒ **Challenge:** tradeoff between extra complexity vs. ECC gain

Codelucida

Outline

1. FAID decoding for LDPC codes

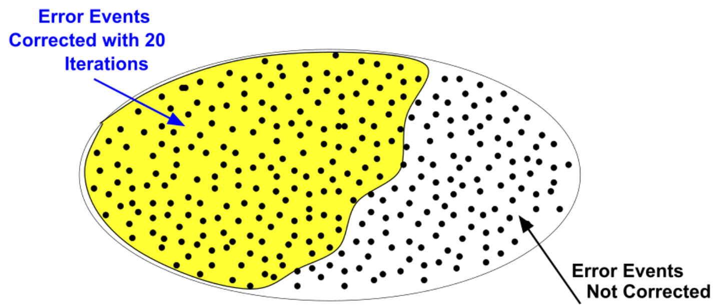
2. Improving Error Correction Performance with **more Iterations**

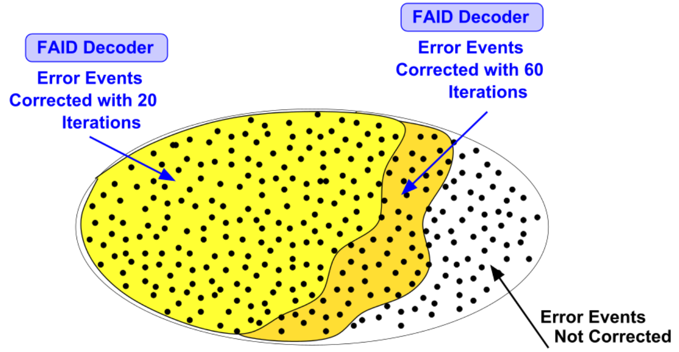
3. Improving Error Correction Performance with **more Precision**

Correcting More Errors with More Iterations

- **Iterative Decoding:** most of the error events (noisy codewords) are corrected within a few iterations

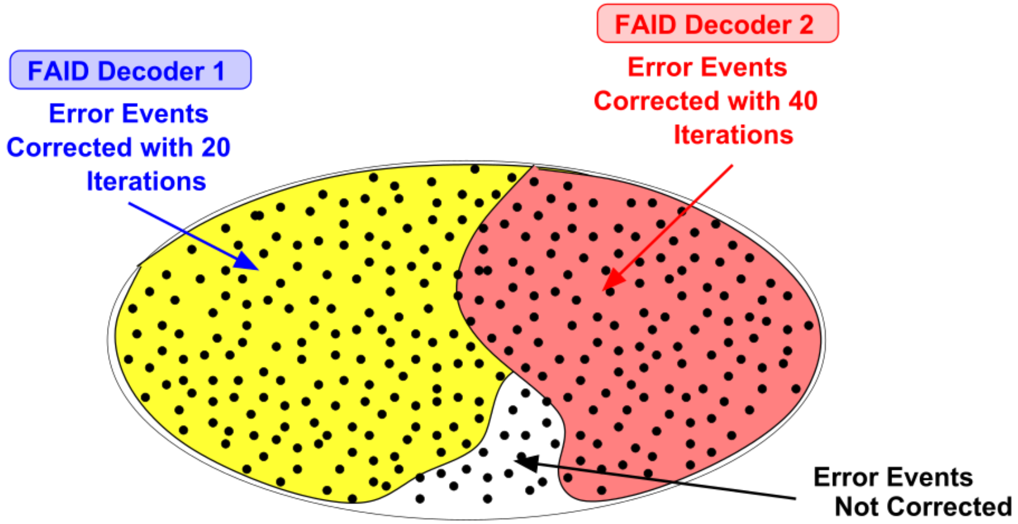
- **Diminishing return** of using larger maximum iteration:
  ECC Gain (10 it. ➝ 20 it.) > ECC Gain (20 it. ➝ 30 it.) > ECC Gain (30 it. ➝ 40 it.) > …


Error Events Corrected with 20 Iterations
Error Events Not Corrected


FAID Decoder
Error Events Corrected with 20 Iterations
FAID Decoder
Error Events Corrected with 60 Iterations
Error Events Not Corrected

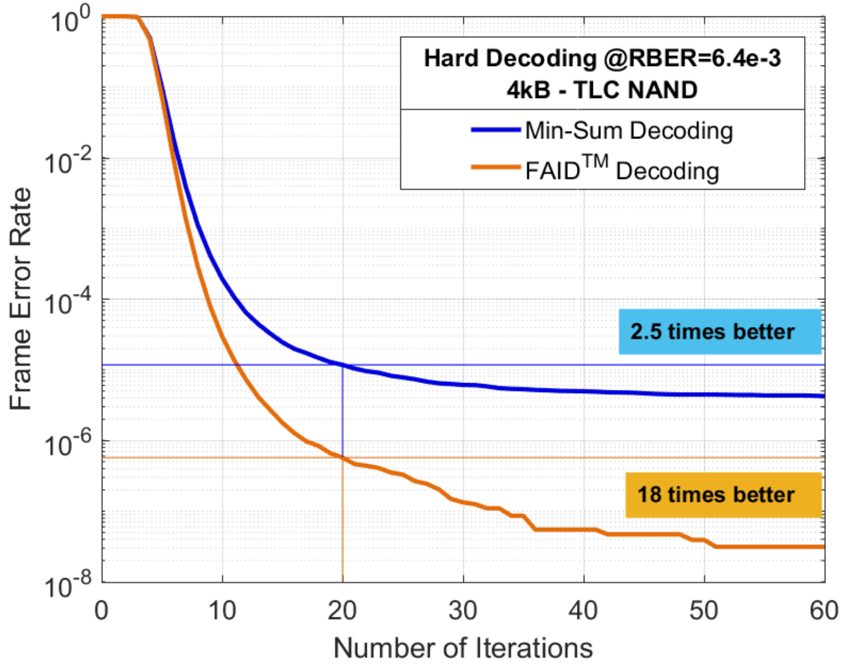
Our solution: FAID Diversity
Correcting More Errors with Multiple FAIDs

- FAID$_1$ is optimized to correct a maximum number of errors within 20 iterations
- FAID$_2$ is optimized to correct a maximum number of errors among those that FAID$_1$ does not correct
- Optimization is performed with **theoretical approaches** (Density Evolution)
- Optimization is universal: does not depend on the LDPC code


FAID Decoder 1
Error Events Corrected with 20 Iterations
FAID Decoder 2
Error Events Corrected with 40 Iterations
Error Events Not Corrected

FAID Diversity performance with more iterations

- Min-Sum does not correct very much more errors after 20 iterations: **diminishing return** of using more iterations
- FAID Diversity experience **less diminishing return** when using more iterations
- FAID with **60 iterations** is **18 times better** than FAID with 20 iterations
- This gain comes at:
  - ➡ **No extra Hardware** complexity
  - ➡ **Negligible Average latency** degradation
  - ➡ Increase of the **worst case latency** only



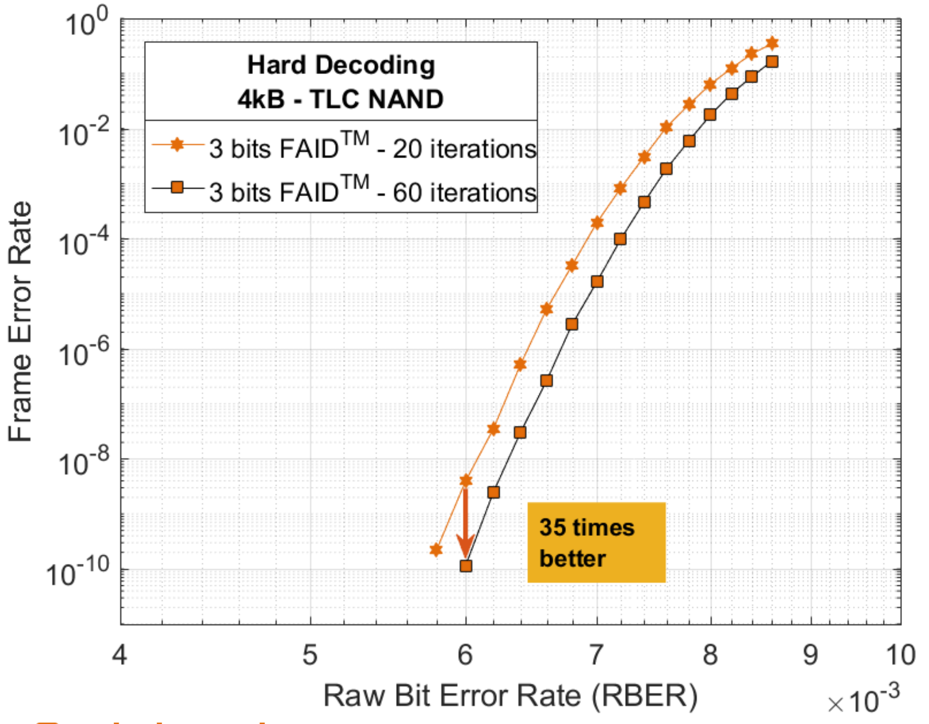
Codelucida

Performance Improvement with more Iterations

- FAID with **60 iterations max.** is **35 times better** than with 20 iterations max.
- Corresponds to **5% gain** in RBER
- Average latency is unchanged:

  20 it: **3.50** average iterations @FER=1e-7

  60 it: **3.52** average iterations @FER=1e-7
- **Benefits:** reduces by **35 times** the need to request soft reads
- Extend the end of life of the Flash



Codelucida

Outline

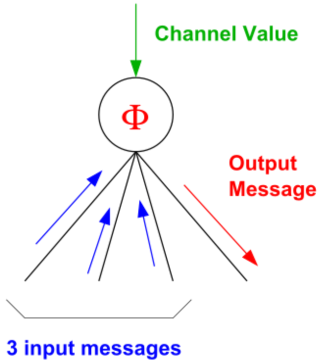
1. FAID decoding for LDPC codes

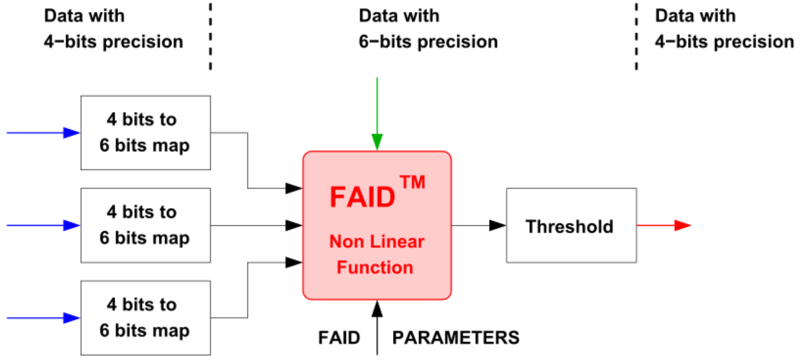
2. Improving Error Correction Performance with **more Iterations**

3. Improving Error Correction Performance with **more Precision**

FAID Variable Node Implementation

- 4-bits messages belonging to $\mathcal{A}$ **= {-7 , -6 , … , -1 , 0 , +1 , … , +6 , +7 }**
- non-linear Boolean function Φ is harder to optimize
- **20% to 30% more Hardware** complexity than 3-bits FAID





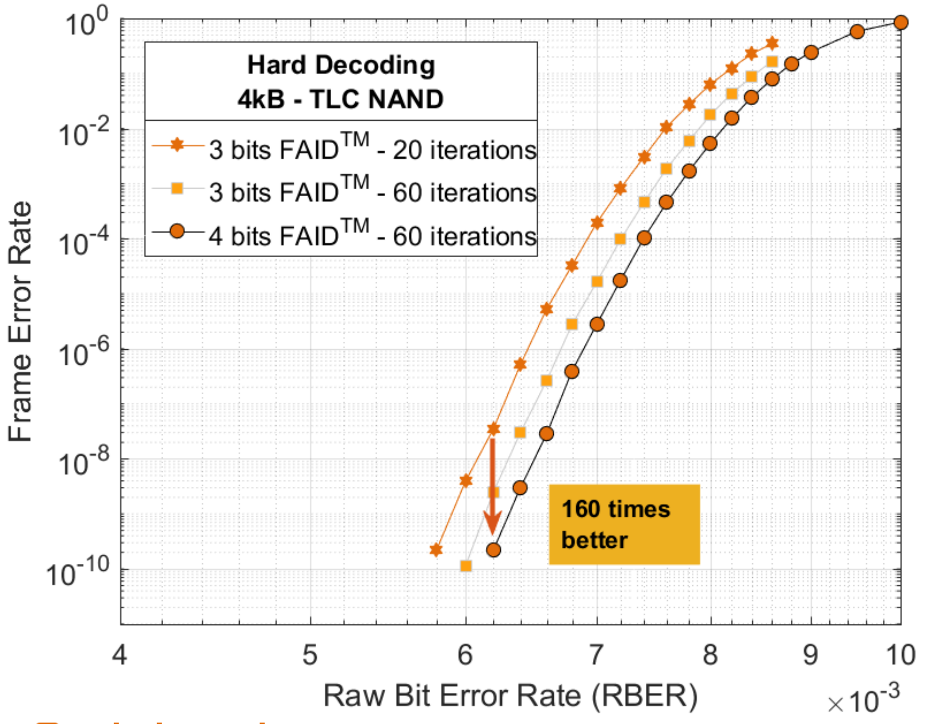
Performance Improvement with more Precision

- 4-bits FAID with 60 it. is **160 times** better than 3-bits FAID with 20 it,
- Corresponds to **9% gain** in RBER
- Average latency is larger:

  20 it: **3.50** average iterations @FER=1e-7

  60 it: **4.70** average iterations @FER=1e-7

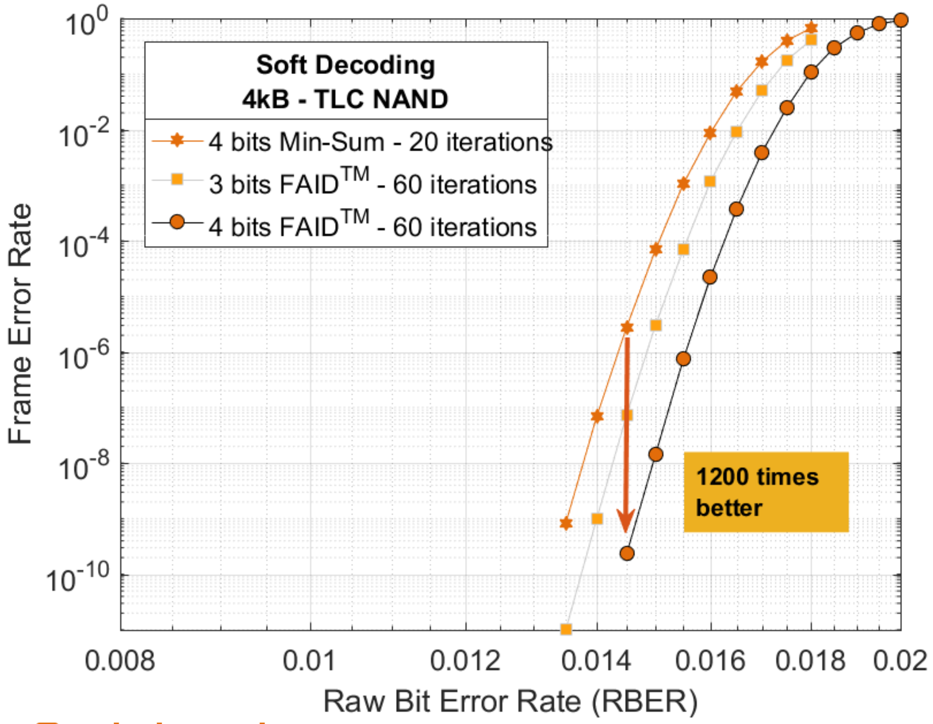
- **Benefits:** reduces by **160 times** the need to request soft reads
- Extend the end of life of the NAND



Codelucida

Performance Improvement for Soft-Decoding

- Soft Decoding
  = 1 hard bit + 1 soft bit
  = 3 NAND reads
- **ECC Gains** brought by extra iterations and 4-bits precision is **even larger for soft decoding**
- 4-bits FAID with 60 it. is **1200 times** better than 4-bits Min-Sum with 20 it,
- Corresponds to **9% gain** in RBER
- **Benefits:** reduces by **1200 times** the need to request more soft reads (5, 7, etc.)



Codelucida

Conclusion

- We showed **ECC performance improvements** for FAID decoding
  - ⇨ **Increasing the maximum Iteration**, not degrading the average latency
  - ⇨ Using **more precision bits** for the messages: 3-bits FAID $\Longrightarrow$ 4-bits FAID
- Helps pushing further **the end of life** of the Flash
- Greatly **limits the number** of soft-read requests

Demonstration at the Booth #952
Fully Flexible FAID Solution on a Xilinx ZU7ev chip