Layer-by-layer Adaptively Optimized ECC for NAND Flash SSD Storing CNN Weights

Keita Mizushina, Toshiki Nakamura and Ken Takeuchi
Chuo University, Tokyo, Japan
Outline

• Introduction
• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)
• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
• Conclusion
Outline

• Introduction

• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)

• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)

• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)

• Conclusion
Concept of This Work

- Train network in data center with GPUs
- Store weight data and infer on Edge devices

Store weight data in SSD

Training (regulate weight)

Data Center

Inference (use weight)

Edge Device

Ex) Self-driving car

Load weight data from SSD

Proposed SSD

SSD-controller

- Proposed LBL-ECC
- Wear-leveling etc...

TLC NAND flash memory
(store weight data)

Errors occur

Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
SegNet Architecture

- SegNet [2] is deep convolutional encoder-decoder architecture for semantic pixel wise labelling

Importance of Weight Data

- Importance of weight data is different among layers

Outer layers are more error acceptable

Inner layers express extracted features

Tolerant error

Low
High

Convolution
Deconvolution

Acceptable bit-error rate, ABER ($\times 10^{-3}$)
Outline

• Introduction
• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)
• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
• Conclusion
Low-Density Parity-Check (LDPC) ECC

- LDPC ECC corrects errors gradually by repeating LDPC ECC decoding [4]

Ex.)
Max iteration = 30 in conventional LDPC decoder

Iteration++

Concept of Proposed LBL-LDPC

- **Layer-by-layer LDPC (LBL-LDPC) adjusts iteration cycle**

Outer layers are more error acceptable

Inner layers express extracted features

Acceptable bit-error rate, ABER ($\times 10^{-3}$)

![Graph showing ABER for different convolution and deconvolution layers.](image)

Strength of proposed LBL-LDPC

- Strong
- Weak

Tolerant error

- High
- Low

Convolution

Deconvolution
Performance of Proposed LBL-LDPC

- LBL-LDPC decreases iteration count without accuracy decline

**Graph:**
- Recognition Success
- Recognition Accuracy
- Measured BER ($\times 10^{-3}$)
- Conventional (Max iteration: 30)
- Proposed LBL-LDPC (15, 15, 30, 30)
- Proposed LBL-LDPC (5, 15, 15, 30)

**Legend:**
- ▲ Conventional (Max iteration: 30)
- ✗ Proposed LBL-LDPC (15, 15, 30, 30)
- ◈ Proposed LBL-LDPC (5, 15, 15, 30)

**Notes:**
- 1Ynm, TLC NAND flash, @150degC, $N_{W/E} = 200$
Performance of Proposed LBL-LDPC

- LBL-LDPC reduces decoding time by 14%
Result of Proposed LBL-LDPC

- Proposed LBL-LDPC decreases decoding time by 14% compared with conventional LDPC ECC

<table>
<thead>
<tr>
<th></th>
<th>Conventional LDPC</th>
<th>Proposed LBL-LDPC (5, 15, 15, 30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC ECC decoding time</td>
<td>96 μs</td>
<td>83 μs</td>
</tr>
</tbody>
</table>

-14%
Outline

• Introduction
• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)
• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
• Conclusion
Conventional Asymmetric Coding (AC)

- At high $V_{TH}$ states, many errors occur
- Asymmetric Coding (AC) [5] increases A-state to minimize total errors

1Ynm, TLC NAND flash, @150degC
Data-retention time = 4 days,
Write/Erase cycles ($N_{W/E}$)=200

Measured BER (a.u.)

$V_{TH}$-state

# of cells

AC $V_{TH}$ distribution

Conventional Asymmetric Coding (AC)

- Flip all bits if “0” is more than “1” and append “1” as flag
Concept of Proposed LBL-AC

- Proposed Layer-by-layer AC (LBL-AC) adjusts AC code length

<table>
<thead>
<tr>
<th>No AC (Random)</th>
<th>AC2 (CL : 4)</th>
<th>AC2 (CL : 8)</th>
<th>AC2 (CL : 16)</th>
<th>AC2 (CL : 32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data overhead</td>
<td>Small</td>
<td>Large</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Code length of LBL-AC</td>
<td>Long</td>
<td>Short</td>
<td>Long</td>
<td>Short</td>
</tr>
</tbody>
</table>

**Measured BER (a.u.)**

1Ynm, TLC NAND flash, @150degC, NW/E = 200

**Proposed LBL-AC**

- Case 1: 6 layers
- Case 2: 4 layers

**Measured BER (a.u.)**

- Convolution
- Deconvolution

Flash Memory Summit 2018
Santa Clara, CA
Performance of Proposed LBL-AC

- Acceptable data-retention time increases by 3.3 times

- Conventional No AC w/ LDPC ECC (iteration = 30)
- Conventional AC2 (CL : 4) w/ LDPC ECC (iteration = 30)
- Proposed LBL-AC case1 (protect central 6 layers) w/ LDPC ECC (iteration = 30)
- Proposed LBL-AC case2 (protect central 4 layers) w/ LDPC ECC (iteration = 30)

- Proposed LBL-LDPC w/o AC w/ 30 iterations
- Proposed LBL-LDPC (5, 15, 15, 30) + Proposed LBL-AC case2

1Ynm, TLC NAND flash, @150degC, NW/E = 200
Performance of Proposed LBL-AC

- LBL-AC reduces data-overhead by 26%

Flash Memory Summit 2018
Santa Clara, CA
Result of Proposed LBL-AC

- Proposed LBL-AC decreases data-overhead by **26%** compared with conventional AC

<table>
<thead>
<tr>
<th></th>
<th>Conventional LDPC ECC</th>
<th>Conventional AC w/ Conventional LDPC</th>
<th>Proposed LBL-AC w/ Conventional LDPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC decoding time</td>
<td>96 μs</td>
<td>96 μs</td>
<td>96 μs</td>
</tr>
<tr>
<td>Flag data overhead rate</td>
<td>0 %</td>
<td>25 %</td>
<td><strong>-26%</strong></td>
</tr>
<tr>
<td>Acceptable data-retention</td>
<td>1.0 day</td>
<td>3.3 days</td>
<td><strong>3.3x</strong></td>
</tr>
<tr>
<td>time</td>
<td></td>
<td></td>
<td>3.3 days</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)
• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
• Conclusion
Summary of Proposed LBL-ECC

- Summarize proposed LBL-ECC method

TLC NAND flash memory (store weight data)

- Wear-leveling
- Bad block management
- Interleaving
- Logical-physical address translation

Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
Layer-by-layer Iteration-Optimized LDPC (LBL-LDPC)
Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
Performance of Proposed LBL-ECC

- Acceptable data-retention time increases by 3.3 times

---

**Graph Details**

- **Conventional LDPC w/o AC w/ 30 iterations**
- **Conventional LDPC w/ AC2 (CL : 4) & 30 iterations**
- **Proposed LBL-LDPC (15, 15, 30, 30)** + **Proposed LBL-AC case 1**
- **Proposed LBL-LDPC (5, 15, 15, 30)** + **Proposed LBL-AC case 2**

**Y-axis**
- Recognition Accuracy
- 1Ynm, TLC NAND flash, @150degC

**X-axis**
- Measured BER (×10⁻³)

**Additional Notes**
- CL : Code Length
  - case 1 : Protect central 6 layers
  - case 2 : Protect central 4 layers

**Measured BER**

- N_{W/E} = 200
Proposed LBL-AC decreases data-overhead by 26% compared with conventional AC.

<table>
<thead>
<tr>
<th></th>
<th>Proposed LBL-LDPC</th>
<th>Conventional LBL-AC w/ Conventional LDPC</th>
<th>Proposed LBL-AC w/ Proposed LBL-AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC decoding time</td>
<td>96 μs</td>
<td>96 μs</td>
<td>83 μs</td>
</tr>
<tr>
<td>Flag data overhead rate</td>
<td>0 %</td>
<td>25 %</td>
<td>19 %</td>
</tr>
<tr>
<td>Acceptable data-retention time</td>
<td>1.0 day</td>
<td>3.3 days</td>
<td>3.3 days</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Proposed Layer-by-layer Iteration-optimized Low-Density Parity-Check Error Correcting Code (LBL-LDPC)
• Proposed Layer-by-layer Code-length Adjusted Asymmetric Coding (LBL-AC)
• Proposed Layer-by-layer Adaptively Optimized Error Correcting Code (LBL-ECC)
• Conclusion
## Conclusion

- Proposed LBL-ECC extends data retention time by 3.3 times

<table>
<thead>
<tr>
<th></th>
<th>Conventional LDPC</th>
<th>Proposed LBL- LDPC (5, 15, 15, 30)</th>
<th>Proposed LBL- LDPC (5, 15, 15, 30) + Proposed LBL- AC Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDPC decoding time</td>
<td>96 µs</td>
<td>83 µs</td>
<td>83 µs</td>
</tr>
<tr>
<td>Flag data overhead rate</td>
<td>0</td>
<td>0</td>
<td>19%</td>
</tr>
<tr>
<td>Acceptable data-retention time</td>
<td>1.0 day</td>
<td>1.0 day</td>
<td>3.3 days</td>
</tr>
</tbody>
</table>
Thank you for your attention

This work was supported by JST CREST Grant Number JPMJCR1532, Japan
Soft-decoding LDPC

- Log-likelihood ratio (LLR) is required for LDPC decoding.

Error Prediction (EP-) LDPC

- EP-LDPC is 7-times faster reads than Soft-decoding LDPC

BER is estimated from some factors of error:
- \( V_{TH} \) information (× 7)
- Inter-cell coupling info.
- Write/Erase cycles \( (N_{W/E}) \)
- Retention time

\[
\begin{align*}
\text{LLR}(1) &= \ln\left\{ \frac{\text{BER}}{1-\text{BER}} \right\} \\
\text{LLR}(0) &= \ln\left\{ \frac{1-\text{BER}}{\text{BER}} \right\}
\end{align*}
\]

Advanced Error Prediction (AEP-) LDPC

- TLC NAND Flash memory is sensitive to program disturb errors
- AEP-LDPC can correct more accurate and efficiency by considering with program disturb

Considering with only data-retention error

Considering with program disturb and data-retention error

References: