Partitioned Real-Time NAND Flash Storage

Katherine Missimer and Rich West
Introduction
Introduction
Introduction

Sensors on Boss

- Velodyne multi-plane lidar
  360° x 28° FOV, 60m

- Continental ISF 177 lidar
  14°, 150m

- BNO 180° FOV
  multi-plane, multi-echo

- Continental ARS 300 radar
  60° / 75°, 40 / 200m

- SICK Scanning Lidar
  90 / 180° FOV, 40m

- Applanix GPS/INS

~16 Sensors total
NAND Flash Memory

- Non-volatility
- Shock resistance
- Low power consumption
- Fast access time
NAND Flash Memory

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-volatility</td>
<td>✅</td>
<td>❌</td>
</tr>
<tr>
<td>Shock resistance</td>
<td>✅</td>
<td>❌</td>
</tr>
<tr>
<td>Low power consumption</td>
<td>✅</td>
<td>❌</td>
</tr>
<tr>
<td>Fast access time</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td>No in-place updates</td>
<td>❌</td>
<td></td>
</tr>
<tr>
<td>Reads &amp; writes operate at different granularity than erasures</td>
<td>❌</td>
<td></td>
</tr>
</tbody>
</table>
SSD Internals
SSD Internals

Flash Block

Page
SSD Internals

Flash Block

Flash Die

Page

Block

Block

Plane 1

Plane 2
SSD Internals
SSD Internals
Observations

NAND Flash Chips

<table>
<thead>
<tr>
<th>CH 1</th>
<th>CH 2</th>
<th>CH 3</th>
<th>CH 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Way 1</td>
<td>Way 2</td>
<td>Way 3</td>
<td>Way 4</td>
</tr>
<tr>
<td>Way 1</td>
<td>Way 2</td>
<td>Way 3</td>
<td>Way 4</td>
</tr>
<tr>
<td>Way 1</td>
<td>Way 2</td>
<td>Way 3</td>
<td>Way 4</td>
</tr>
<tr>
<td>Way 1</td>
<td>Way 2</td>
<td>Way 3</td>
<td>Way 4</td>
</tr>
</tbody>
</table>

Write latency for 4 pages

<table>
<thead>
<tr>
<th></th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same chip</td>
<td>9.70 msec</td>
</tr>
<tr>
<td>Way interleaving</td>
<td>2.90 msec</td>
</tr>
<tr>
<td>Channel striping</td>
<td>2.37 msec</td>
</tr>
</tbody>
</table>

Erasure latency for 4 blocks

<table>
<thead>
<tr>
<th></th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same chip</td>
<td>16.2 msec</td>
</tr>
<tr>
<td>Way interleaving</td>
<td>4.06 msec</td>
</tr>
<tr>
<td>Channel striping</td>
<td>4.06 msec</td>
</tr>
</tbody>
</table>

Read latency for 4 pages

<table>
<thead>
<tr>
<th></th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same chip</td>
<td>1.44 msec</td>
</tr>
<tr>
<td>Way interleaving</td>
<td>1.25 msec</td>
</tr>
<tr>
<td>Channel striping</td>
<td>0.382 msec</td>
</tr>
</tbody>
</table>
PaRT-FTL

<table>
<thead>
<tr>
<th>Channel 1</th>
<th>Way 1</th>
<th>Way 2</th>
<th>Way 3</th>
<th>Way 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel 1</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR(D1,D2,D3)</td>
</tr>
<tr>
<td>Channel 2</td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR(D4,D5,D6)</td>
</tr>
<tr>
<td>Channel 3</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR(D7,D8,D9)</td>
</tr>
<tr>
<td>Channel 4</td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR(D10,D11,D12)</td>
</tr>
</tbody>
</table>
PaRT–FTL

Way 1 | Way 2 | Way 3 | Way 4
---|---|---|---
Channel 1 | D1 | D2 | D3 | XOR(D1,D2,D3)
Channel 2 | D4 | D5 | D6 | XOR(D4,D5,D6)
Channel 3 | D7 | D8 | D9 | XOR(D7,D8,D9)
Channel 4 | D10 | D11 | D12 | XOR(D10,D11,D12)
PaRT-FTL

Channel 1: D1, D2, XOR(D1,D2,D3)
Channel 2: D4, D5, XOR(D4,D5,D6)
Channel 3: D7, D8, XOR(D7,D8,D9)
Channel 4: D10, D11, XOR(D10,D11,D12)

Way 1
Way 2
Way 3
Way 4

Write
Read
PaRT-FTL

Way 1: D1, D4, D7, D10
Way 2: D2, D5, D8, D11
Way 3: D3, D6, D9, D12
Way 4: XOR, XOR, XOR

Buffered pages:

- D1
- D2
- D3
- XOR

Time:

- t₀

Write: D1, D4, D7, D10
Read: D2, D5, D8, D11

XOR: D3, D6, D9, D12
PaRT-FTL

Buffered pages

Time

D1  D2  D3  XOR
PaRT-FTL

Buffered pages

Time

Way 1

Way 2

Way 3

Way 4

D1 D2 D3 XOR

D1 D2 D3 XOR

D1 D2 D3 XOR

D1 D2 D3 XOR

read

write

read
PaRT-FTL

Way 1 | Way 2 | Way 3 | Way 4
---|---|---|---
D1 | D2 | D3 | XOR
D4 | D5 | D6 | XOR
D7 | D8 | D9 | XOR
D10 | D11 | D12 | XOR

Buffered pages

Time:

- $t_0$: D1, D4, D7, D10
- $t_1$: D1, D4, D7, D10

D1: Invalid
D2: Invalid
D3: Invalid
XOR: Invalid

Legend:
- Orange: Invalid
- Blue: Read
- Purple: Write
- Green: XOR
PaRT-FTL

Buffered pages

Time

Way 1

Way 2

Way 3

Way 4

D1 D2

D3 D4

D5 D6

D7 D8

D9 D10

D11 D12

D1 D2

D3 D4

D5 D6

D7 D8

D9 D10

D11 D12

read

write

read

Invalid

Invalid

XOR

XOR

XOR

XOR

XOR

XOR

XOR

XOR

D1 D2

D3 D4

D5 D6

D7 D8

D9 D10

D11 D12

read

write

read
PaRT-FTL

Buffered pages

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1</strong></td>
<td><strong>D2</strong></td>
<td><strong>D3</strong></td>
<td><strong>XOR</strong></td>
</tr>
<tr>
<td>Invalid</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time

- **t₀**: D1 → D4 → D7 → D10
  - Way 1: D1, D4, D7, D10
  - Way 2: D2, D5, D8, D11
  - Way 3: D3, D6, D9, D12
  - Way 4: XOR, XOR, XOR

- **t₁**: D1 → D4 → D7 → D10
  - Way 1: D1, D4, D7, D10
  - Way 2: D2, D5, D8, D11
  - Way 3: D3, D6, D9, D12
  - Way 4: XOR, XOR, XOR

- **t₂**: D1 → D4 → D7 → D10
  - Way 1: D1, D4, D7, D10
  - Way 2: D2, D5, D8, D11
  - Way 3: D3, D6, D9, D12
  - Way 4: XOR, XOR, XOR

- **t₃**: D1 → D4 → D7 → D10
  - Way 1: D1, D4, D7, D10
  - Way 2: D2, D5, D8, D11
  - Way 3: D3, D6, D9, D12
  - Way 4: XOR, XOR, XOR

**Read**

- D1, D2, D3, D4, D5, D6

**Write**

- D7, D8, D9, D10, D11, D12
## PaRT–FTL

<table>
<thead>
<tr>
<th>Time</th>
<th>Way 1</th>
<th>Way 2</th>
<th>Way 3</th>
<th>Way 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₀</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR</td>
</tr>
<tr>
<td>t₁</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR</td>
</tr>
<tr>
<td>t₂</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR</td>
</tr>
<tr>
<td>t₃</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR</td>
</tr>
<tr>
<td>t₄</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>XOR</td>
</tr>
<tr>
<td></td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
<td>XOR</td>
</tr>
</tbody>
</table>

### Buffered pages

```
D1 | D2 | D3 | XOR
---|----|----|-----
```

**Write:**
- D1
- D7

**Read:**
- D2
- D8
- D11
Real-Time Task Model

Parameters for a task: \[ (r, T_r), (w, T_w) \]
Real-Time Task Model

Parameters for a task: \[ (r, T_r), (w, T_w) \]

- \# pages in read request
- read period
- \# pages in write request
- write period

A task does not account for the CPU computation time.

These tasks exist on the flash translation layer and utilize the NAND bus.
Real-Time Task Model

Parameters for a task: \[ (r, T_r), (w, T_w) \]

Read capacity:

\[ C_r = r \cdot (t_r + t_d) \]

- \# pages in read request
- time to read a page
- time to decode a page

Write capacity:

\[ C_w = \left[ \frac{w}{|F_w|} \right] \cdot t_w \]
Real-Time Task Model

Parameters for a task: \( [(r, T_r), (w, T_w)] \)

Read capacity:
\[
C_r = r \cdot (t_r + t_d)
\]

Write capacity:
\[
C_w = \left[ \frac{w}{F_w} \right] \cdot t_w
\]

# pages in write request
# write chips
time to write a page
Real-Time Task Model

Channel 1
- D1
- D2
- D3
- XOR(D1,D2,D3)

Channel 2
- D4
- D5
- D6
- XOR(D4,D5,D6)

Channel 3
- D7
- D8
- D9
- XOR(D7,D8,D9)

Channel 4
- D10
- D11
- D12
- XOR(D10,D11,D12)

$k = 12$  \quad m = 4
Real-Time Task Model

Encoding task when \( w > 0 \):

\[
C_{en} = m \cdot P \cdot t_{en} + \left[ \frac{m \cdot P}{F_w} \right] \cdot t_w
\]

- \( k = 12 \)
- \( m = 4 \)
Real-Time Task Model

Encoding task when $w > 0$:

$$C_{en} = m \cdot P \cdot t_{en} + \left\lceil \frac{m \cdot P}{|F_w|} \right\rceil \cdot t_w$$

$$T_{en} = T_w \cdot \frac{k \cdot P}{w}$$

$k = 12$  \hspace{1cm}  $m = 4$
Real-Time Task Model

Garbage collection task when $\nu > 0$:
1. Write 3 flash pages of Logical Page Numbers (LPN): 0, 1, 2

---

**Page-level Mapping Table**

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4000</td>
</tr>
<tr>
<td>1</td>
<td>4001</td>
</tr>
<tr>
<td>2</td>
<td>4002</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

---

**Block 1000 (data)**

<table>
<thead>
<tr>
<th>PPN</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
</tr>
<tr>
<td>4001</td>
<td>y</td>
</tr>
<tr>
<td>4002</td>
<td>z</td>
</tr>
<tr>
<td>4003</td>
<td></td>
</tr>
</tbody>
</table>
NAND Flash Memory Garbage Collection

1. Write 3 flash pages of Logical Page Numbers (LPN): 0, 1, 2
2. Update LPN=0

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4003</td>
</tr>
<tr>
<td>1</td>
<td>4001</td>
</tr>
<tr>
<td>2</td>
<td>4002</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Block 1000 (data)

<table>
<thead>
<tr>
<th>PPN</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>✗</td>
</tr>
<tr>
<td>4001</td>
<td>y</td>
</tr>
<tr>
<td>4002</td>
<td>z</td>
</tr>
<tr>
<td>4003</td>
<td>x'</td>
</tr>
</tbody>
</table>
NAND Flash Memory Garbage Collection

1. Write 3 flash pages of Logical Page Numbers (LPN): 0, 1, 2
2. Update LPN=0
3. GC triggered to reclaim Block 1000

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4003</td>
</tr>
<tr>
<td>1</td>
<td>4001</td>
</tr>
<tr>
<td>2</td>
<td>4002</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Block 1000 (data)

<table>
<thead>
<tr>
<th>PPN</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
</tr>
<tr>
<td>4001</td>
<td>y</td>
</tr>
<tr>
<td>4002</td>
<td>z</td>
</tr>
<tr>
<td>4003</td>
<td>x'</td>
</tr>
</tbody>
</table>

Block 2000 (free)

<table>
<thead>
<tr>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
</tr>
<tr>
<td>8001</td>
</tr>
<tr>
<td>8002</td>
</tr>
<tr>
<td>8003</td>
</tr>
</tbody>
</table>
1. Write 3 flash pages of Logical Page Numbers (LPN): 0, 1, 2
2. Update LPN=0
3. GC triggered to reclaim Block 1000
4. Copy valid pages in victim block to a free block

<table>
<thead>
<tr>
<th>Page-level Mapping Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPN</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Block 1000 (data)

<table>
<thead>
<tr>
<th>PPN</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
</tr>
<tr>
<td>4001</td>
<td>y</td>
</tr>
<tr>
<td>4002</td>
<td>z</td>
</tr>
<tr>
<td>4003</td>
<td>x'</td>
</tr>
</tbody>
</table>

Block 2000 (free)

<table>
<thead>
<tr>
<th>PPN</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>y</td>
</tr>
<tr>
<td>8001</td>
<td>z</td>
</tr>
<tr>
<td>8002</td>
<td>x'</td>
</tr>
<tr>
<td>8003</td>
<td></td>
</tr>
</tbody>
</table>
1. Write 3 flash pages of Logical Page Numbers (LPN): 0, 1, 2

2. Update LPN=0

3. GC triggered to reclaim Block 1000

4. Copy valid pages in victim block to a free block

5. Erase victim block
Real-Time Task Model

Garbage collection task when $w > 0$:

Over-provisioning

- Logical Address Space
- FTL address mapping
- Physical Address Space
Garbage collection task when $w > 0$:

- **Over-provisioning**
  - Logical Address Space
  - *FTL address mapping*
  - Physical Address Space

- **GC victim block**
  - Invalid page
  - Invalid page
Real-Time Task Model

Garbage collection task when $w > 0$:

\[ C_{gc} = \text{time spent copying valid pages and erasing the victim block} \]

\[ T_{gc} = \text{time before the next block needs to be reclaimed} \]
Admission Control

**Read set:**
\[
\frac{t_r}{\text{min}_T r} + \sum_{i=1}^{n} \frac{C_r^i}{T_r^i} \leq 1
\]

**Write set:**
\[
\frac{t_e}{\text{min}_T} + \sum_{i=1}^{n} \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1
\]
Admission Control

Read set:

\[ \frac{t_r}{\text{min}_T T_r} + \sum_{i=1}^{n} \frac{C_r^i}{T_r^i} \leq 1 \]

Write set:

\[ \frac{t_e}{\text{min}_T T} + \sum_{i=1}^{n} \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1 \]
Admission Control

**Read set:**

\[
\frac{t_r}{\text{min}_r T_r} + \sum_{i=1}^{n} \frac{C_r^i}{T_r^i} \leq 1
\]

**Write set:**

**time to erase a block**

\[
\frac{t_e}{\text{min}_r T} + \sum_{i=1}^{n} \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1
\]

**minimum period in all write, encoding and GC tasks**
Admission Control Simulations

# of task sets: 500

# of tasks per task set: 10

Each task makes 1-page read and 3-page write requests per period

Periods are calculated based on generated utilizations

![Graph showing schedulable tasksets versus utilization]
OpenSSD Cosmos Board
8 tasks

4 tasks make write requests:
12-page writes every 60 msec

4 tasks make read requests:
3-page reads every 15 msec
Experimental Results: WAO-GC

8 tasks

4 tasks make write requests: 12-page writes every 60 msec

4 tasks make read requests: 3-page reads every 15 msec
Experimental Results

![Graph showing page latency comparison between PaRTFTL and WAO-GC.]

- **Write Max**
- **Write Avg**
- **Read Max**
- **Read Avg**
Conclusion

Contributions of PaRT-FTL:

- An FTL design that takes advantage of internal parallelism in SSDs
- A real-time task model for read and write requests on multiple flash chips
- Bounded and low-latency read requests that are not blocked by write requests or garbage collection