

# Partitioned Real-Time NAND Flash Storage

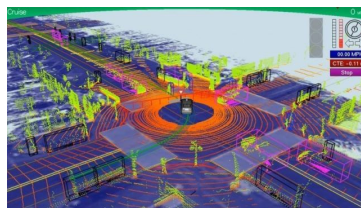
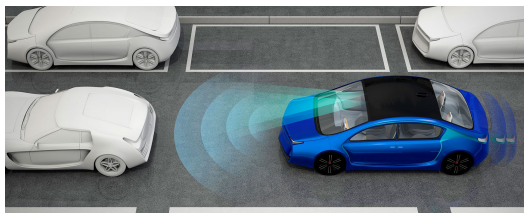
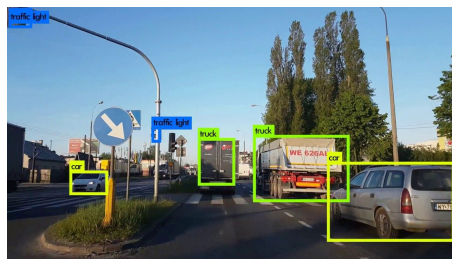
Katherine Missimer and Rich West

**BOSTON**  
**UNIVERSITY**

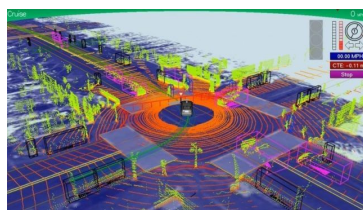
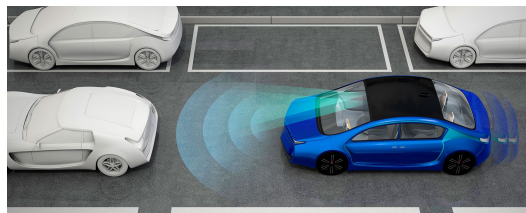
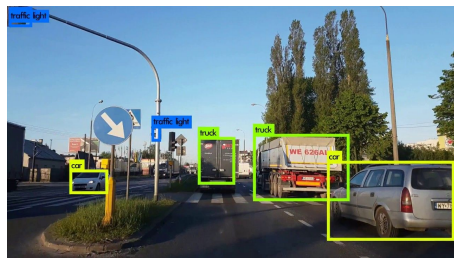
# Introduction



# Introduction



# Introduction



## Sensors on Boss



Velodyne multi-plane lidar  
360°x26° FOV, 60m



Applanix GPS/INS



Continental ISF 172 lidar  
14°, 150m



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



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



Continental ARS 300 radar  
60°/17°, 60°/200m

~16 Sensors total

# NAND Flash Memory

- ✓ Non-volatility
- ✓ Shock resistance
- ✓ Low power consumption
- ✓ Fast access time

# NAND Flash Memory



Non-volatility



Shock resistance



Low power consumption



Fast access time



No in-place updates



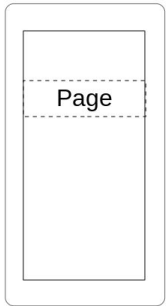
Reads & writes operate at different granularity than erasures

# SSD Internals



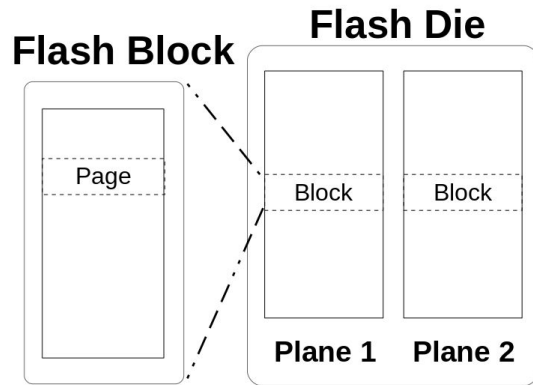
# SSD Internals

## Flash Block

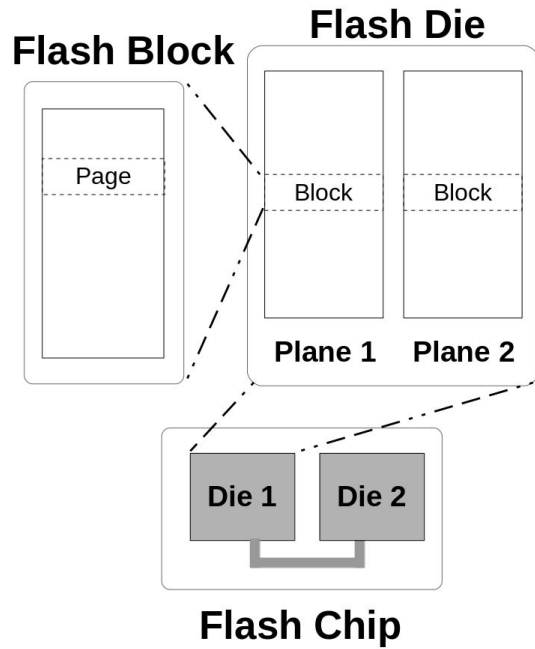




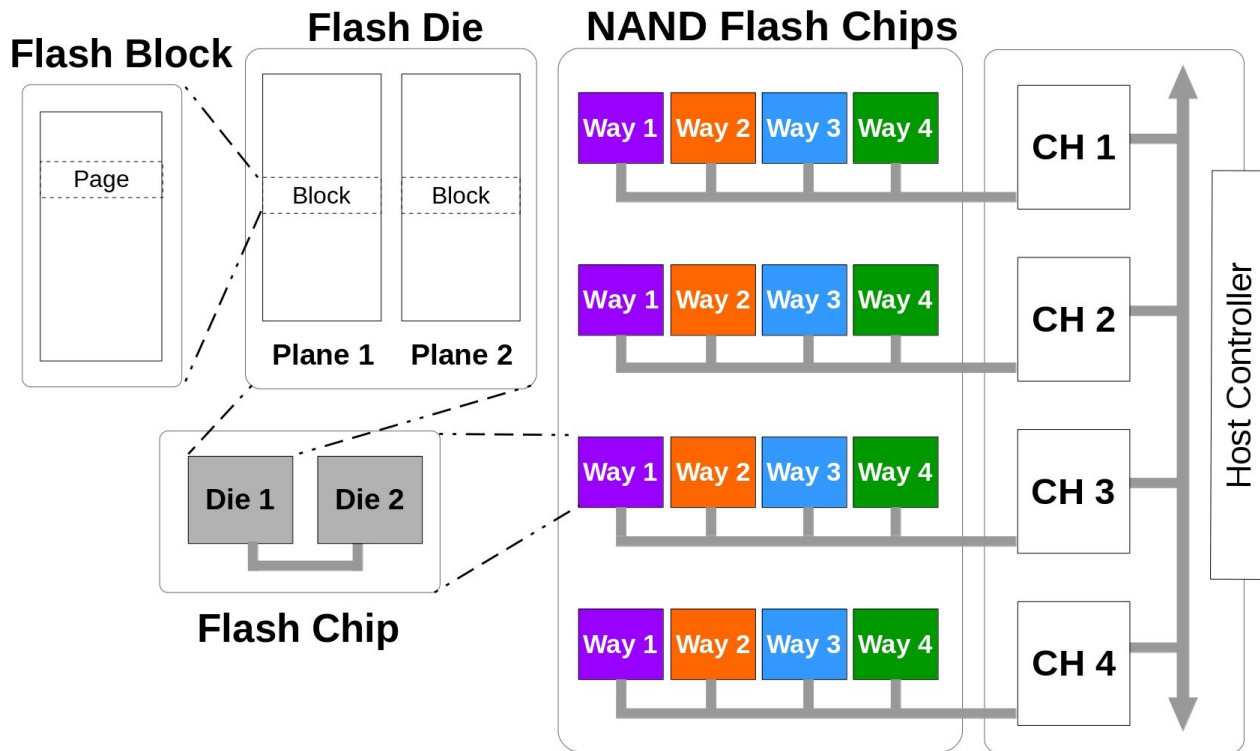
# SSD Internals



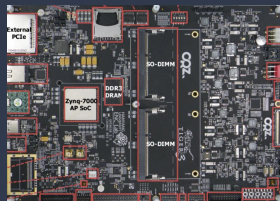
# SSD Internals



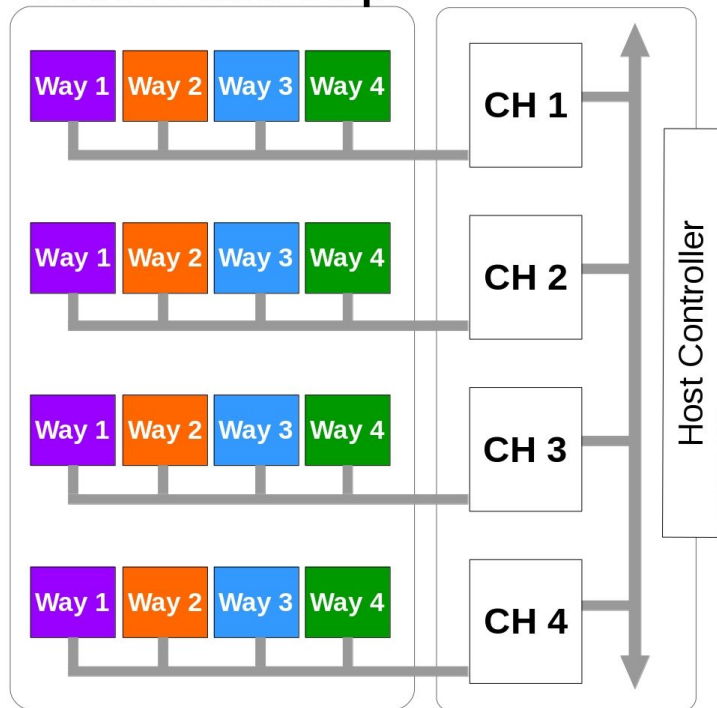
# SSD Internals



# Observations



## NAND Flash Chips



## Write latency for 4 pages

Same chip	9.70 msec
Way interleaving	2.90 msec
Channel striping	2.37 msec

## Erase latency for 4 blocks

Same chip	16.2 msec
Way interleaving	4.06 msec
Channel striping	4.06 msec

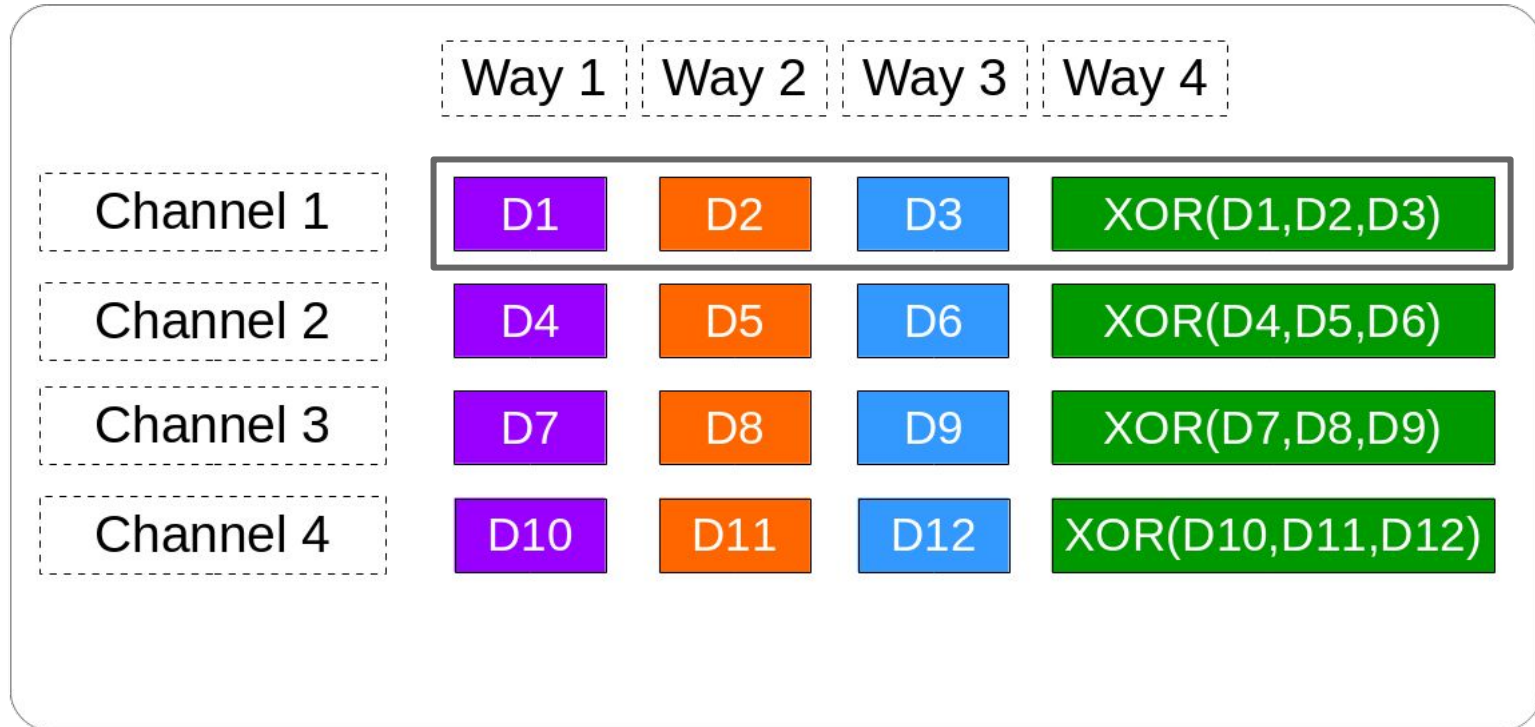
## Read latency for 4 pages

Same chip	1.44 msec
Way interleaving	1.25 msec
Channel striping	0.382 msec

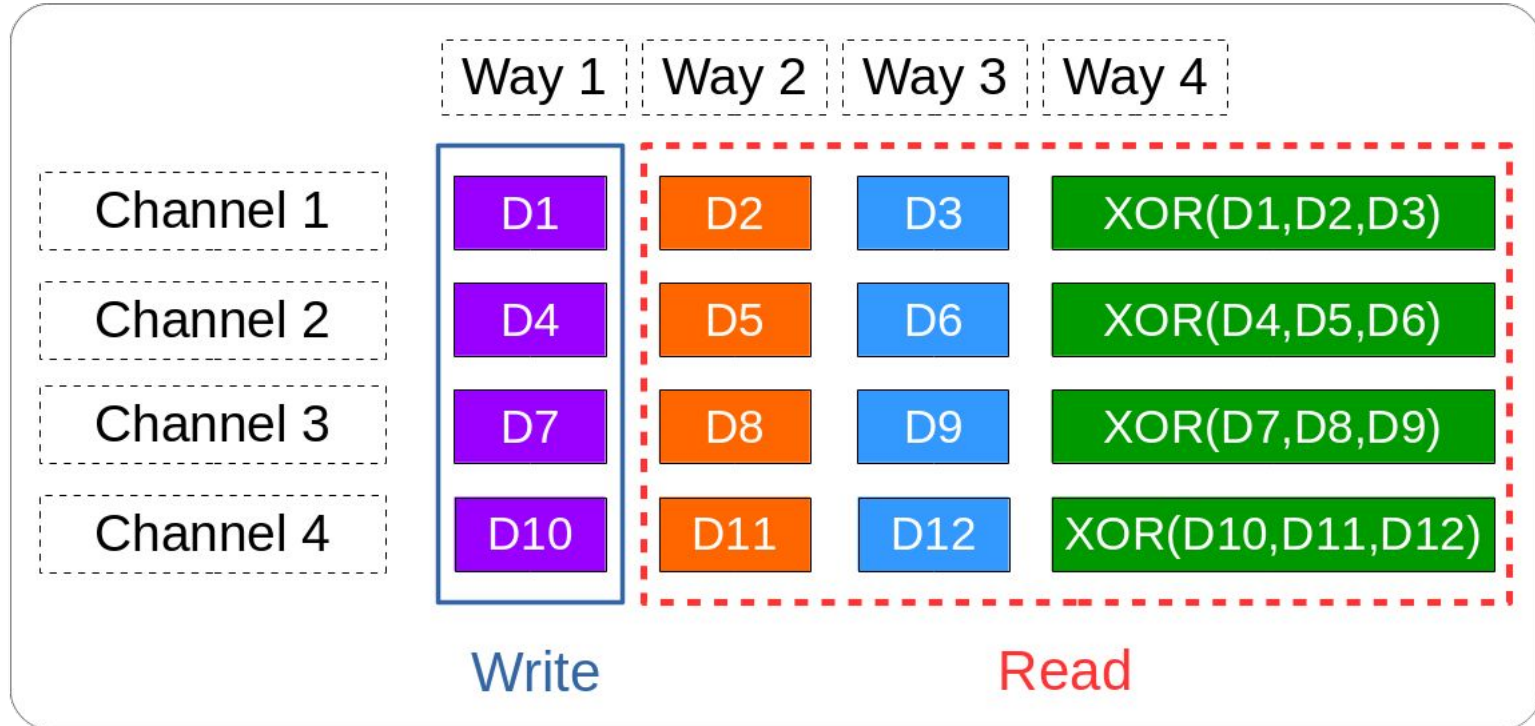
# PaRT-FTL



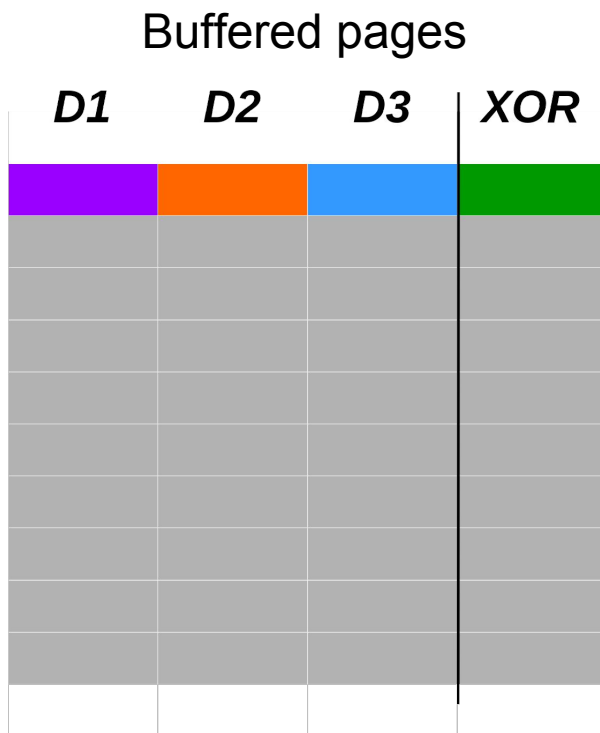
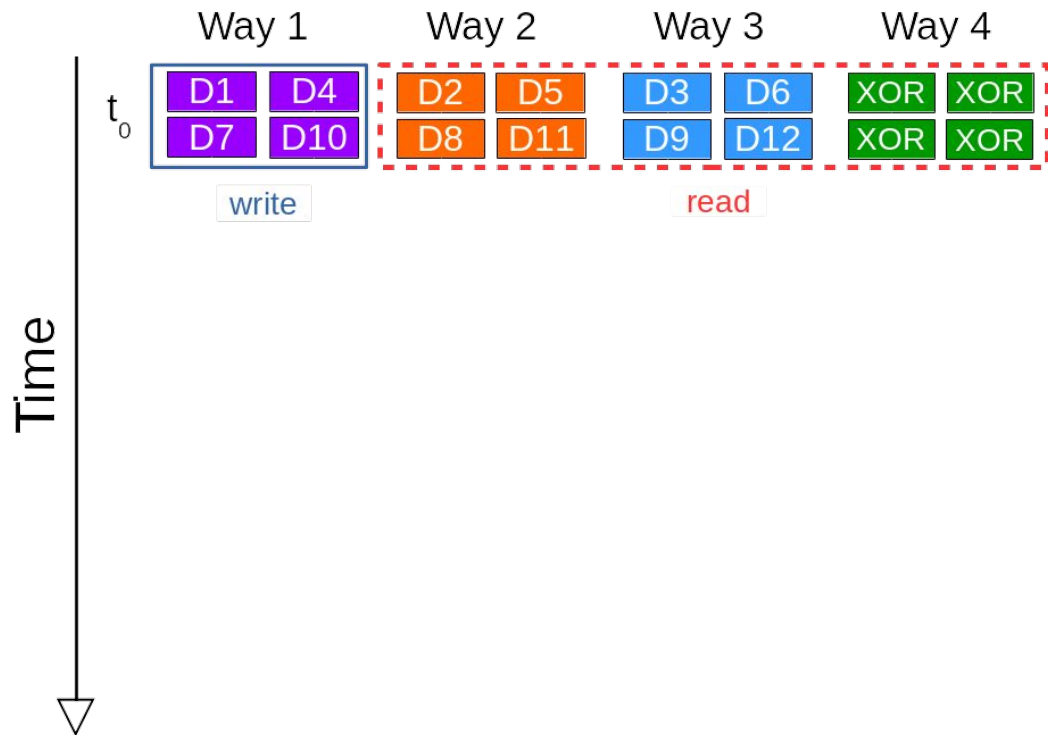
# PaRT-FTL



# PaRT-FTL

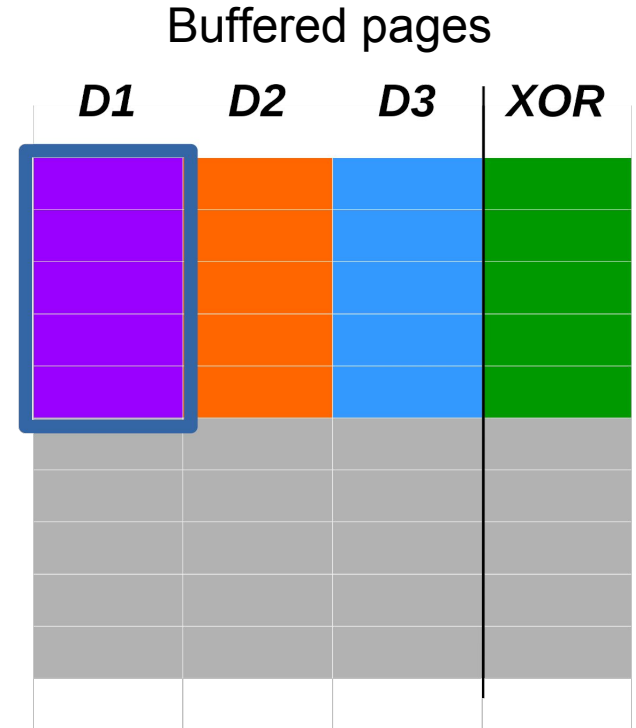
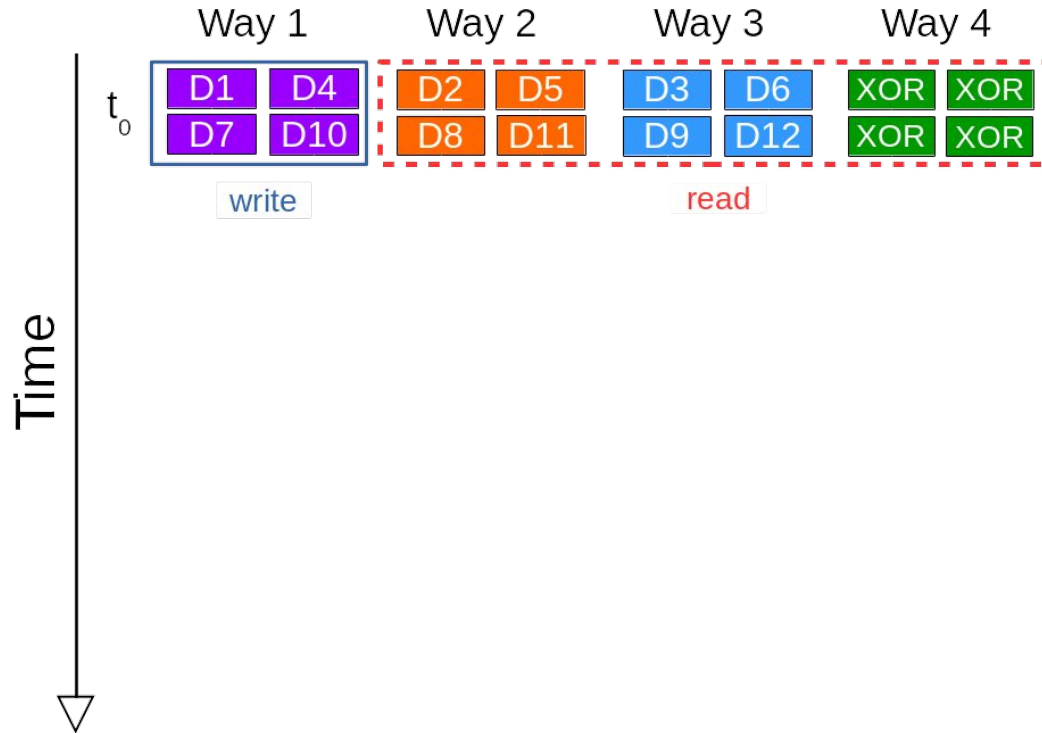


# PaRT-FTL

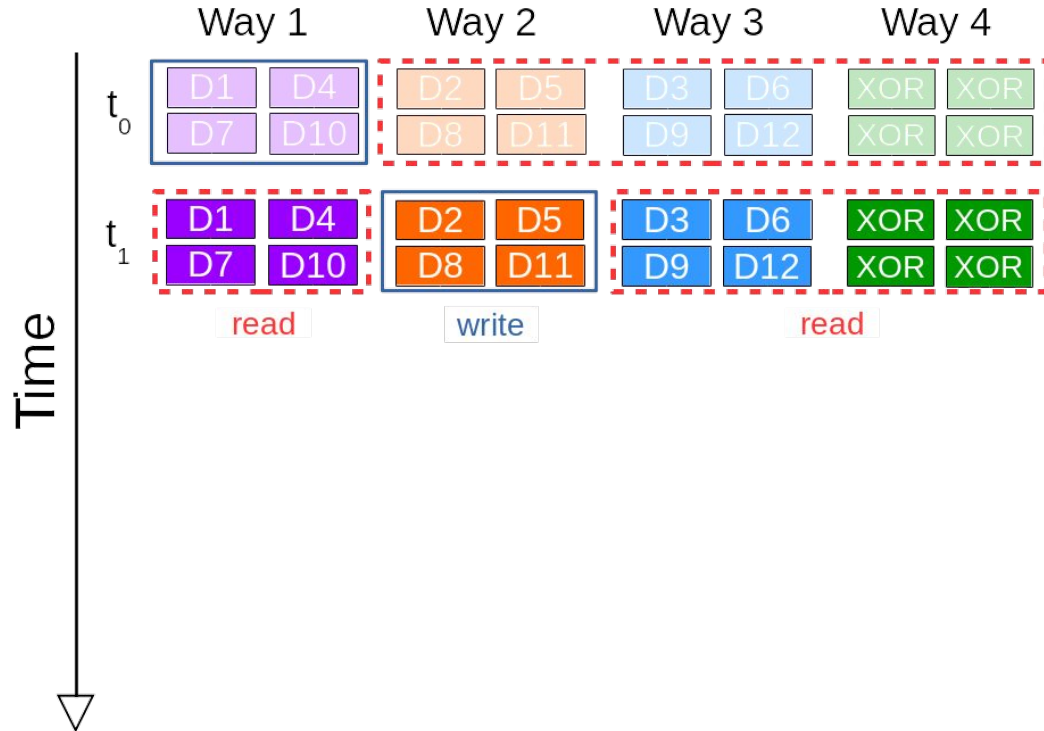




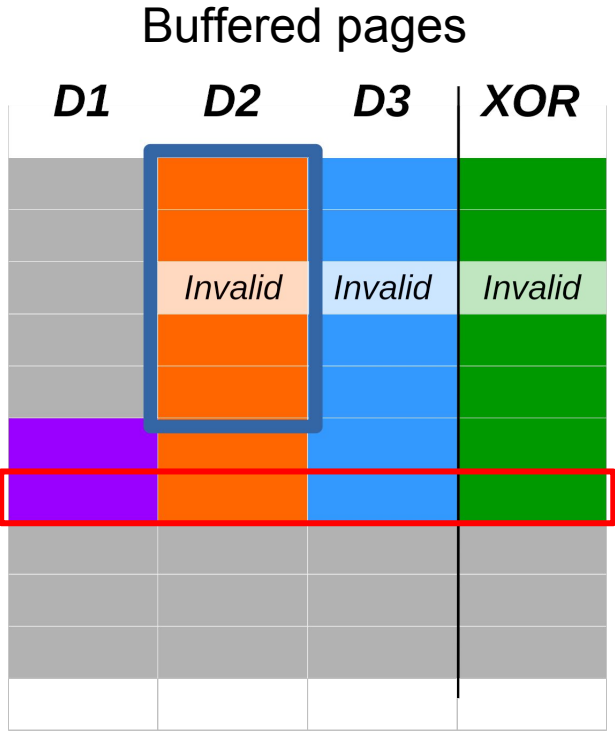
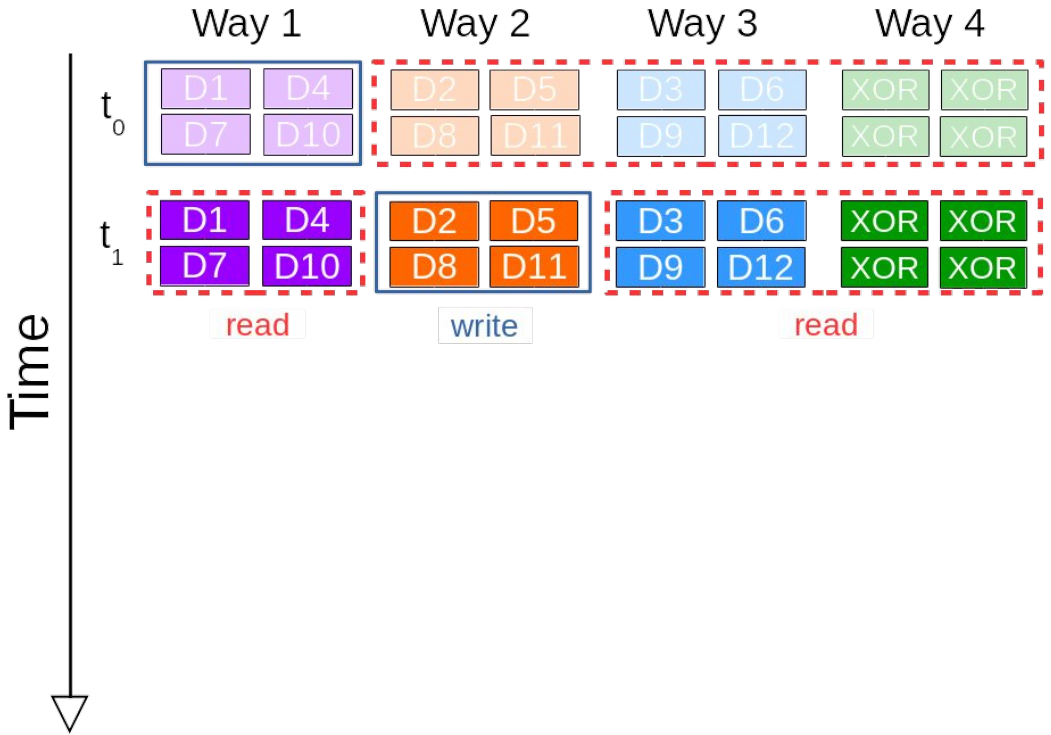
# PaRT-FTL



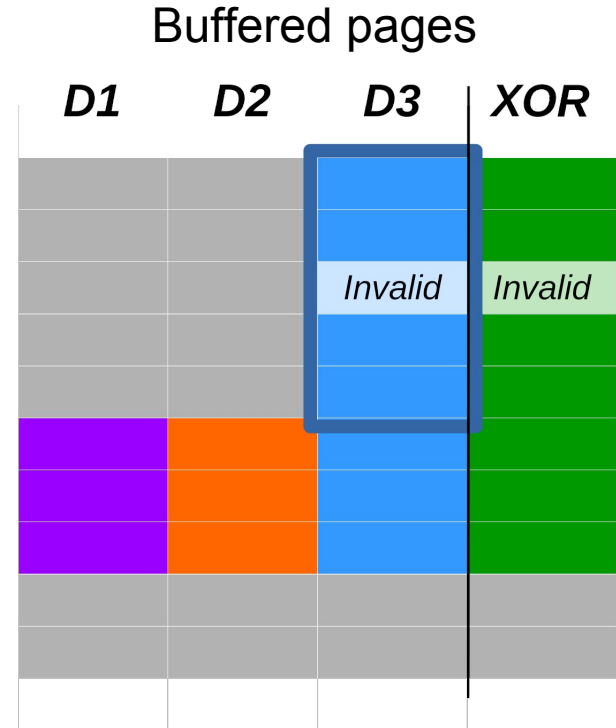
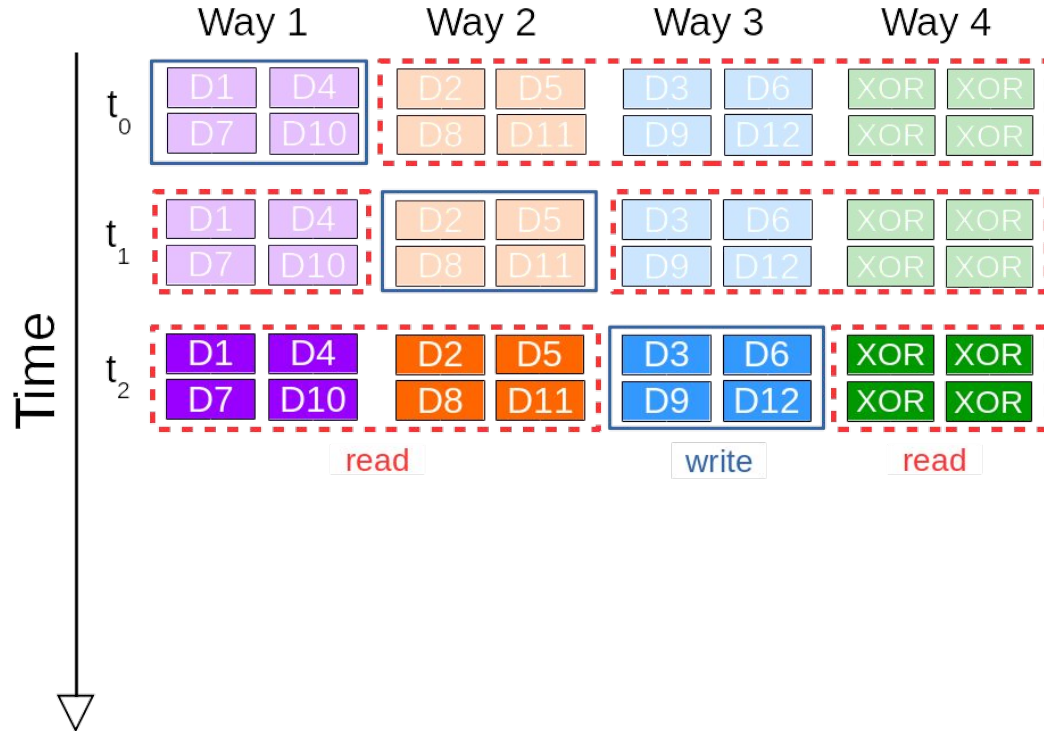
# PaRT-FTL



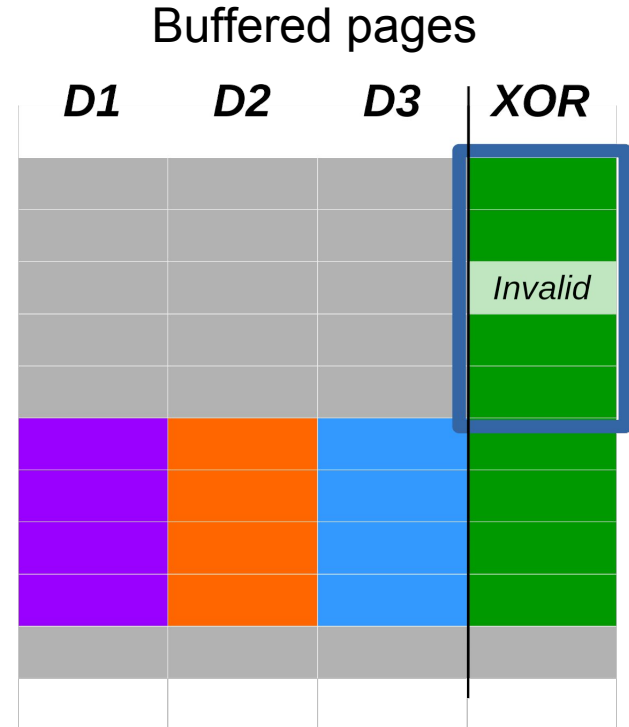
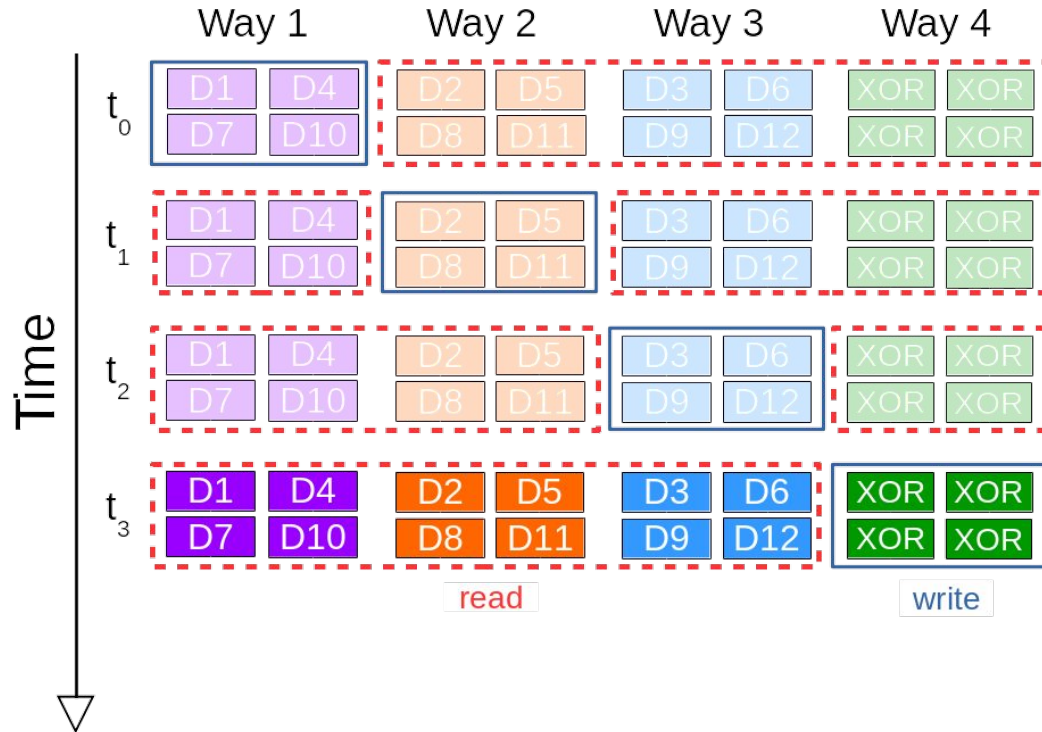
# PaRT-FTL



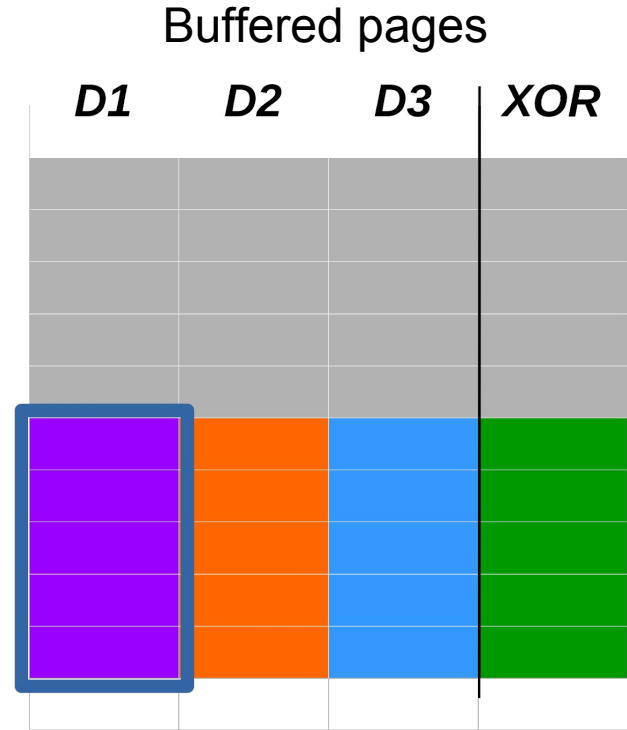
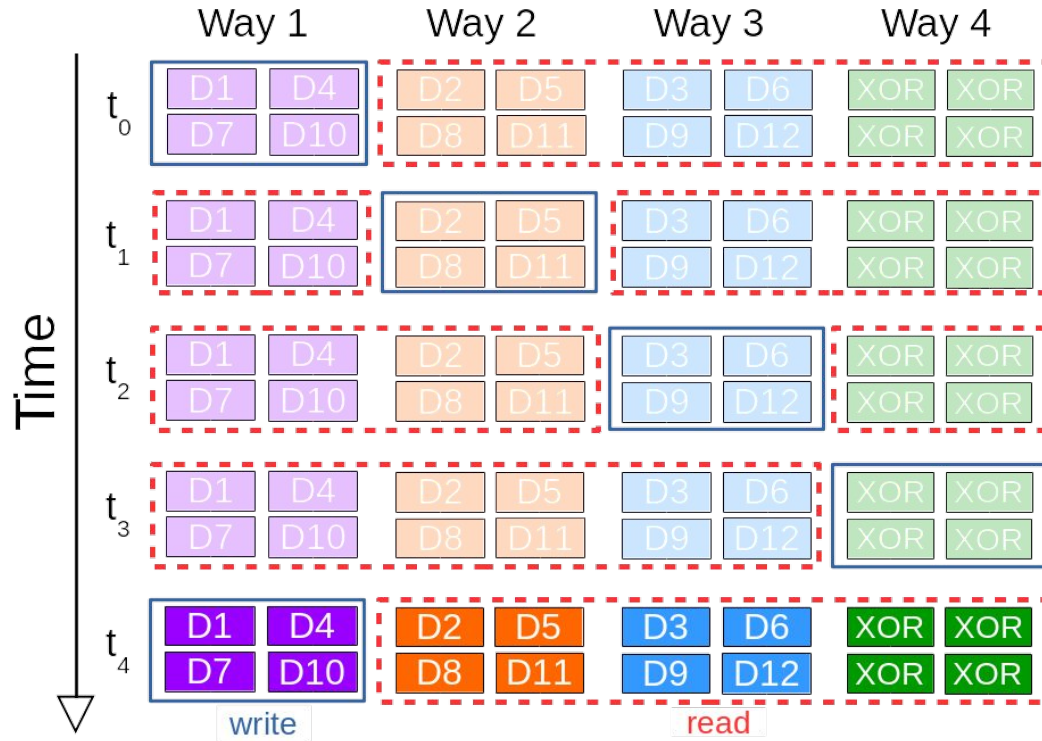
# PaRT-FTL



# PaRT-FTL



# PaRT-FTL



# Real-Time Task Model

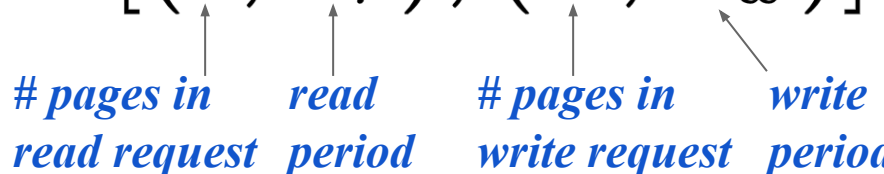
Parameters for a task:  $[(r, T_r), (w, T_w)]$

*# pages in read request*    *read period*    *# pages in write request*    *write period*

# 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:

Write capacity:

$$C_r = r \cdot (t_r + t_d)$$

$$C_w = \left\lceil \frac{w}{|F_w|} \right\rceil \cdot t_w$$

*# pages in  
read request*

*time to  
read a page*

*time to  
decode a page*

# 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:

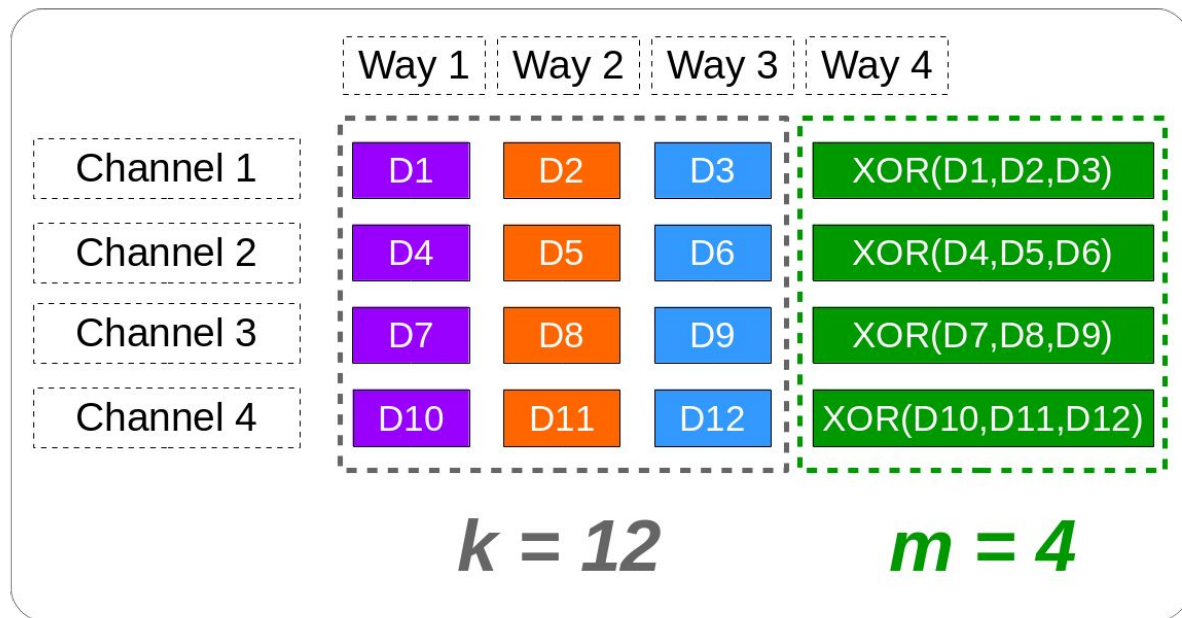
$$C_w = \left[ \frac{w}{|F_w|} \right] \cdot t_w$$

*# pages in  
write request*

*# write  
chips*

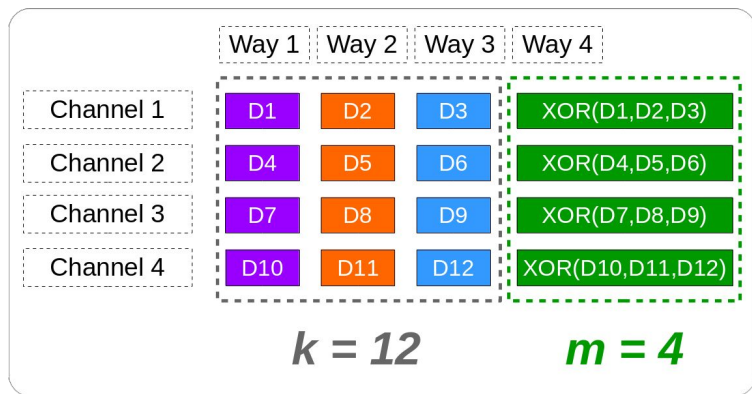
*time to write  
a page*

# Real-Time Task Model



# Real-Time Task Model

Encoding task when  $w > 0$ :

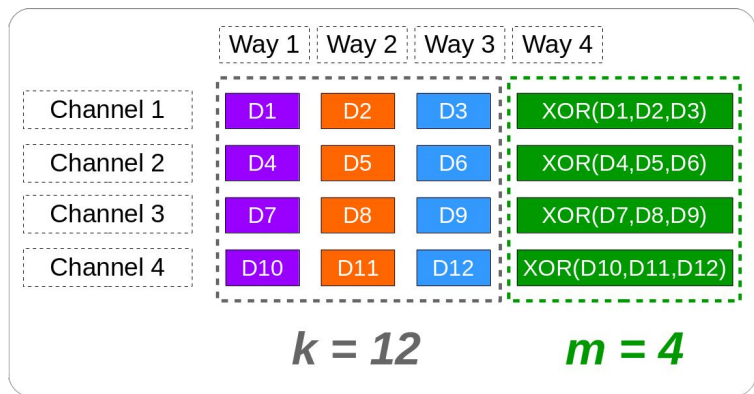


$$C_{en} = m \cdot P \cdot t_{en} + \left\lceil \frac{m \cdot P}{|F_w|} \right\rceil \cdot t_w$$

*time to encode a page*
*# pages per block*
*# write chips*
*time to write a page*

# Real-Time Task Model

Encoding task when  $w > 0$ :



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

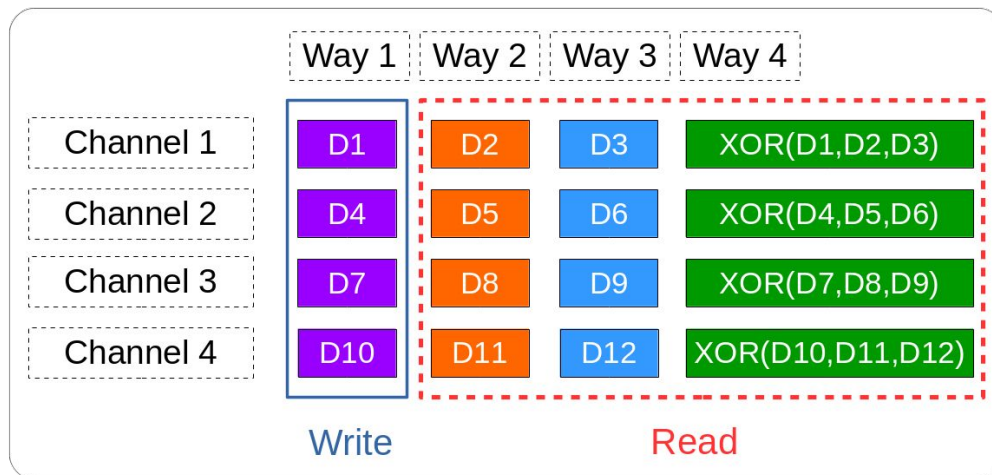
$$T_{en} = T_w \cdot \frac{k \cdot P}{w}$$

$\uparrow$  write period
 $\uparrow$  # pages in write request

$\nwarrow$  # pages per block

# Real-Time Task Model

Garbage collection task when  $w > 0$ :



# NAND Flash Memory Garbage Collection

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

Page-level Mapping Table

LPN	PPN
0	4000
1	4001
2	4002
3	

Block 1000 (data)

PPN	data
4000	x
4001	y
4002	z
4003	

# NAND Flash Memory Garbage Collection

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

Page-level Mapping Table

LPN	PPN
0	4003
1	4001
2	4002
3	

Block 1000 (data)

PPN	data
4000	<del>x</del>
4001	y
4002	z
4003	x'



# 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

Page-level Mapping Table

LPN	PPN
0	4003
1	4001
2	4002
3	

Block 1000 (data)

PPN	data
4000	<del>x</del>
4001	y
4002	z
4003	x'

Block 2000 (free)

PPN	data
8000	
8001	
8002	
8003	

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

Page-level Mapping Table

LPN	PPN
0	8002
1	8000
2	8001
3	

Block 1000 (data)

PPN	data
4000	<del>x</del>
4001	y
4002	z
4003	x'

Block 2000 (free)

PPN	data
8000	y
8001	z
8002	x'
8003	

# 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
4. Copy valid pages in victim block to a free block
5. Erase victim block

Page-level Mapping Table

LPN	PPN
0	8002
1	8000
2	8001
3	

Block 1000 (data)

PPN	data
4000	
4001	
4002	
4003	

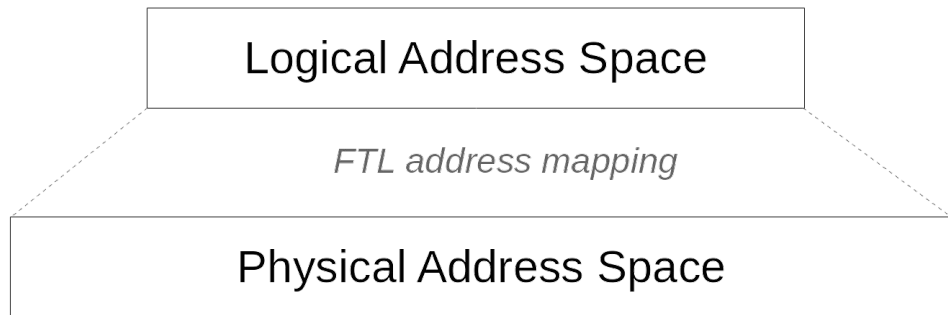
Block 2000 (free)

PPN	data
8000	y
8001	z
8002	x'
8003	

# Real-Time Task Model

Garbage collection task when  $w > 0$ :

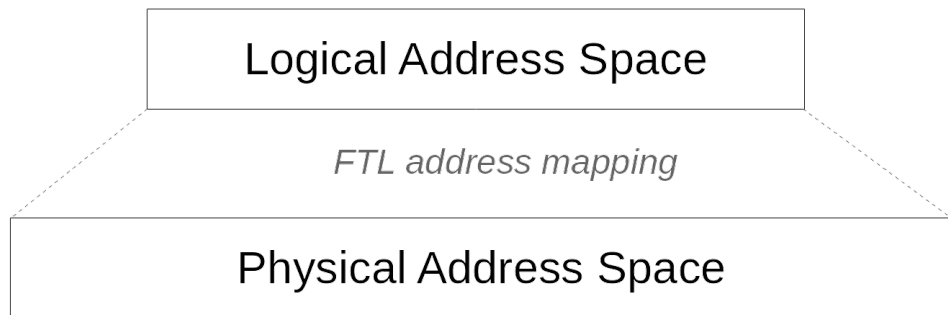
Over-provisioning



# Real-Time Task Model

## Garbage collection task when $w > 0$ :

Over-provisioning



GC victim block



# Real-Time Task Model

Garbage collection task when  $w > 0$ :

$C_{gc}$  = time spent copying valid pages and erasing the victim block

$T_{gc}$  = time before the next block needs to be reclaimed

# Admission Control

*Read set:*

$$\frac{t_r}{\min\_T_r} + \sum_{i=1}^n \frac{C_r^i}{T_r^i} \leq 1$$

*Write set:*

$$\frac{t_e}{\min\_T} + \sum_{i=1}^n \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1$$

# Admission Control

*Read set:*

*time to  
read a page*

$$\frac{t_r}{\min\_T_r} + \sum_{i=1}^n \frac{C_r^i}{T_r^i} \leq 1$$

*Write set:*

*minimum read period*

$$\frac{t_e}{\min\_T} + \sum_{i=1}^n \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1$$



# Admission Control

*Read set:*

$$\frac{t_r}{\min\_T_r} + \sum_{i=1}^n \frac{C_r^i}{T_r^i} \leq 1$$

*Write set:* *time to  
erase a block*

$$\frac{t_e}{\min\_T} + \sum_{i=1}^n \left( \frac{C_w^i}{T_w^i} + \frac{C_{en}^i}{T_{en}^i} + \frac{C_{gc}^i}{T_{gc}^i} \right) \leq 1$$

*minimum period in all write, encoding and GC tasks*

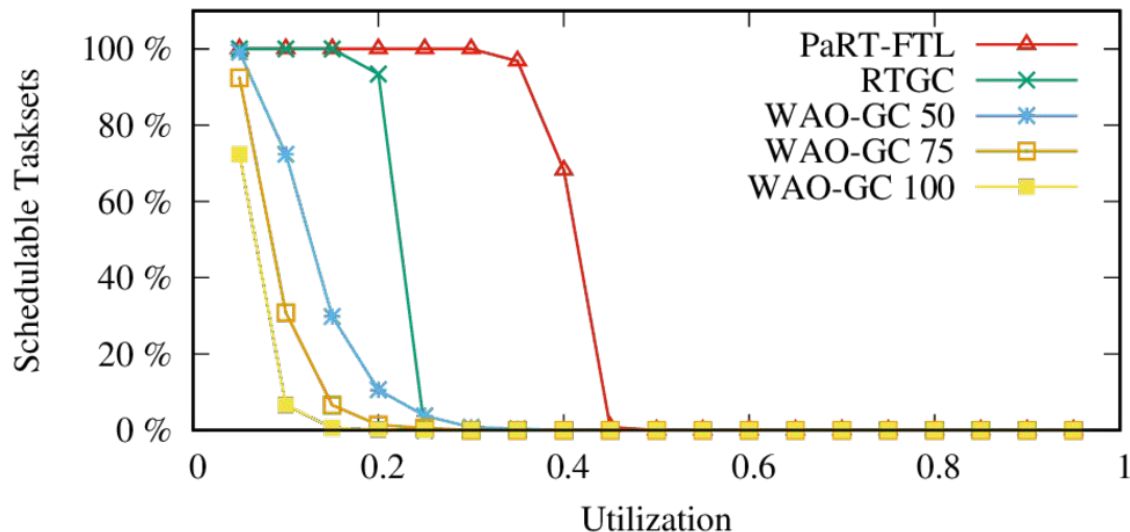
# Admission Control Simulations

# of task sets: 500

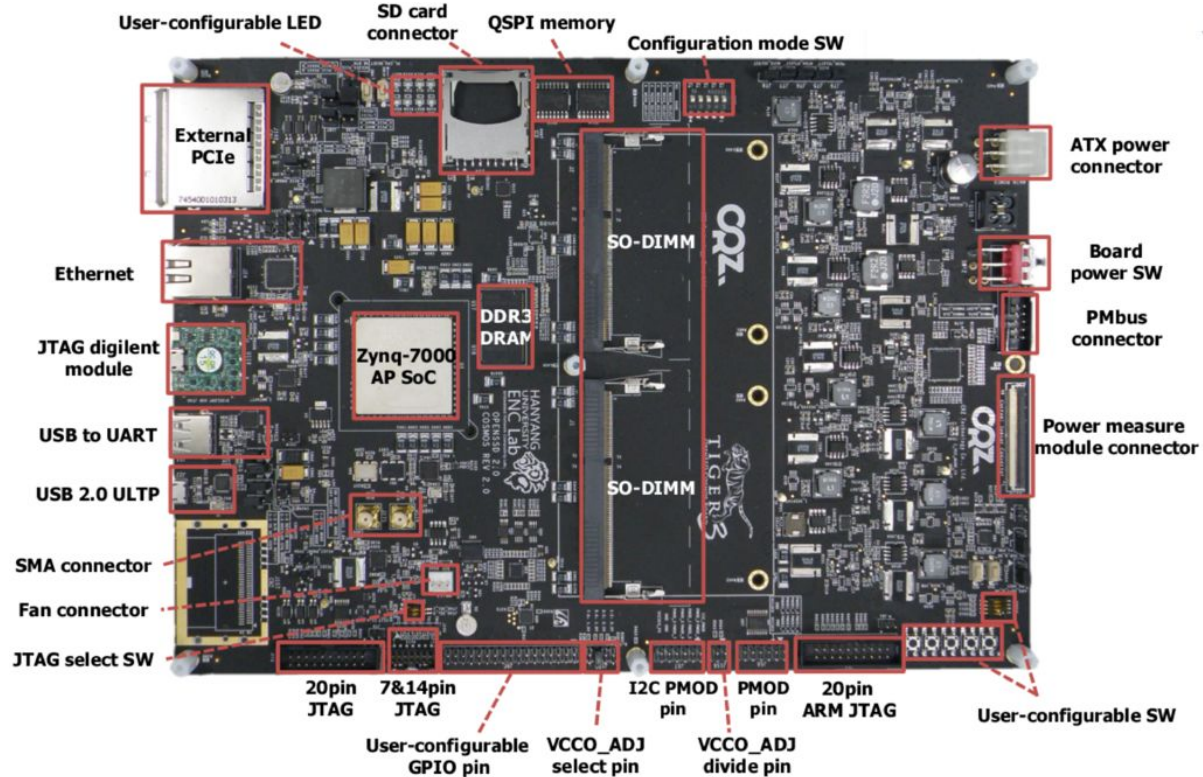
# of tasks per task set: 10

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

Periods are calculated based on  
generated utilizations



# OpenSSD Cosmos Board



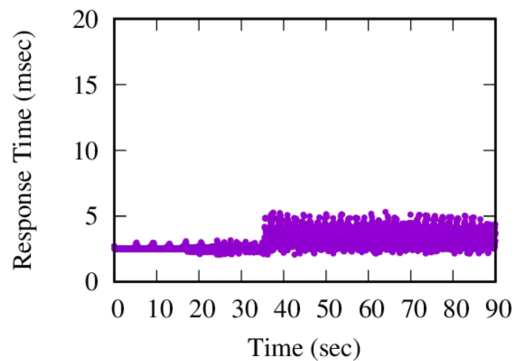
# Experimental Results: PaRT-FTL

8 tasks

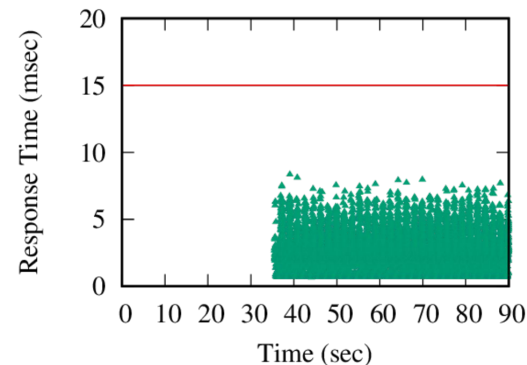
4 tasks make write requests:  
12-page writes every 60 msec

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

Write requests



Read requests

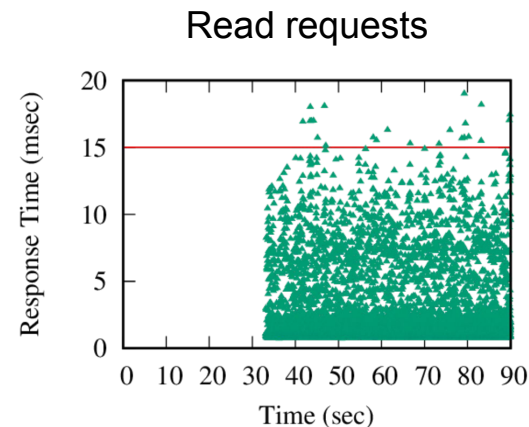
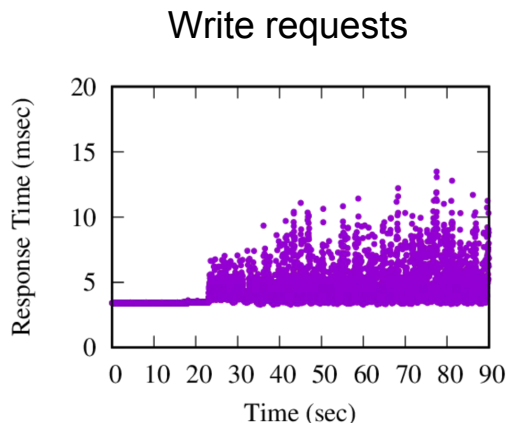


# Experimental Results: WAO-GC

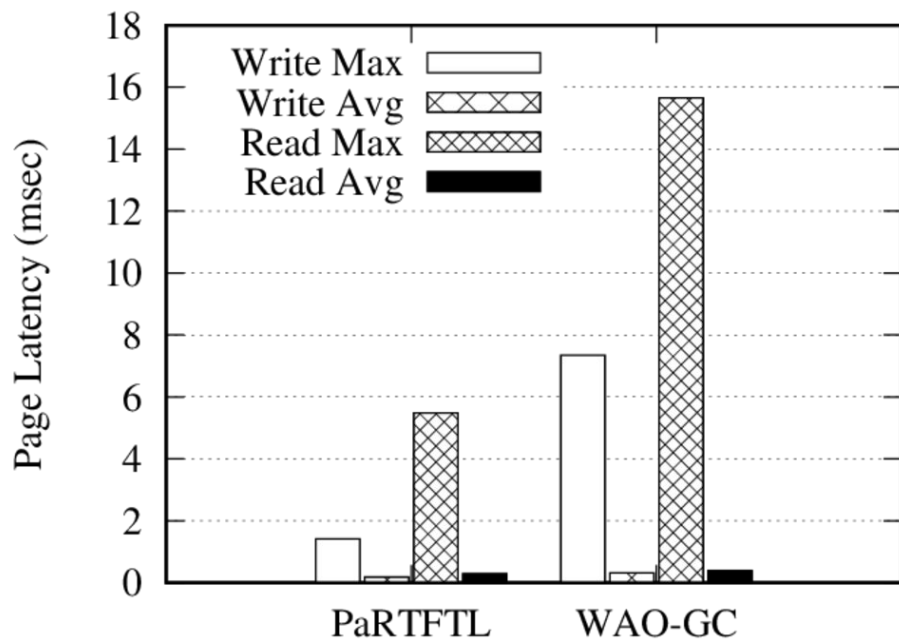
8 tasks

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

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



# Experimental Results



# 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