Could We Make SSDs Self-Healing?

Tong Zhang

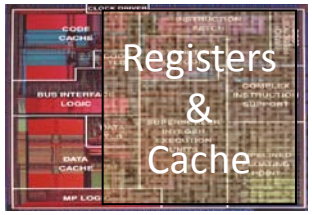
Electrical, Computer and Systems Engineering Department

Rensselaer Polytechnic Institute

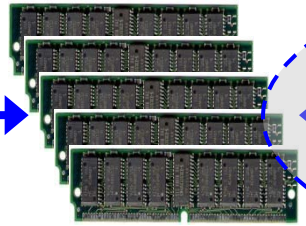
Google/Bing: “tong rpi”

Introduction and Motivation

Hot Topic

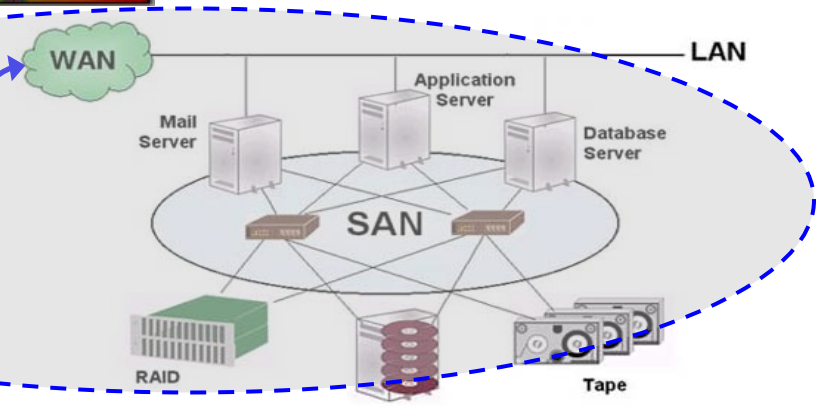


CPU

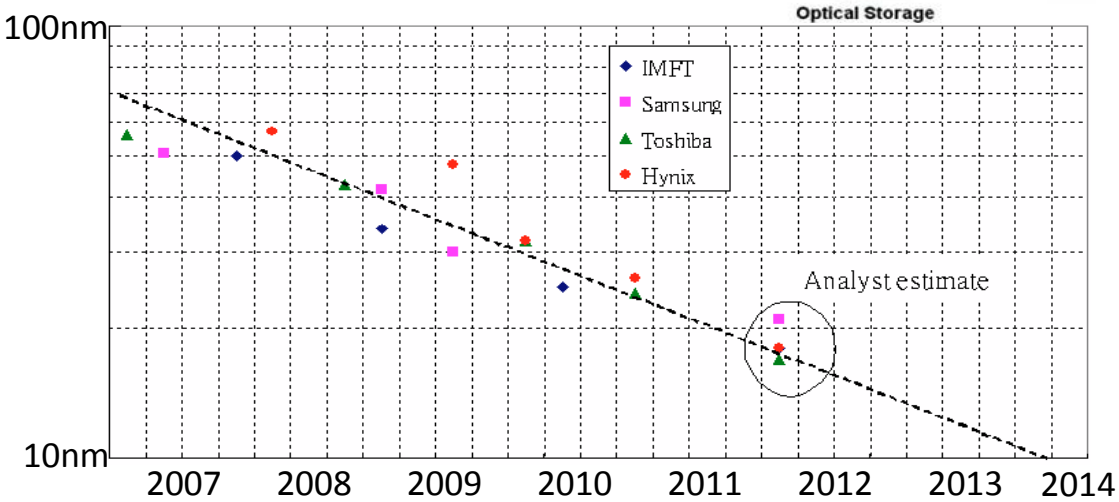


Main memory

Hard disk

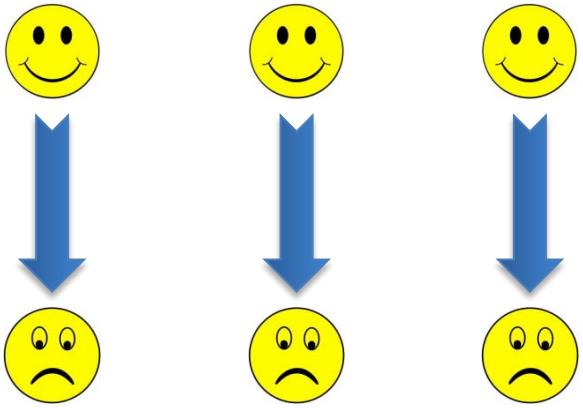


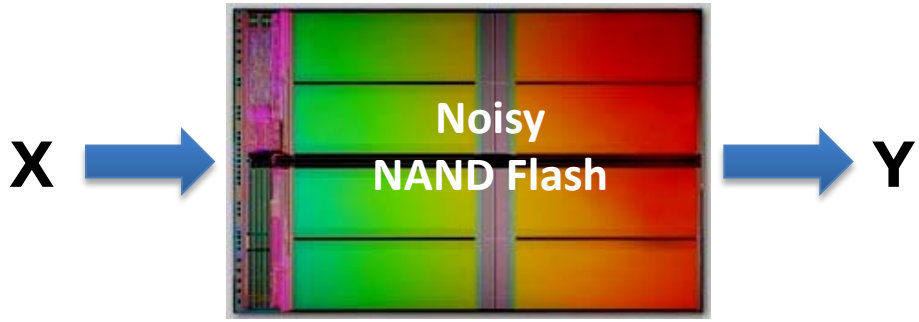
Bit cost reduction



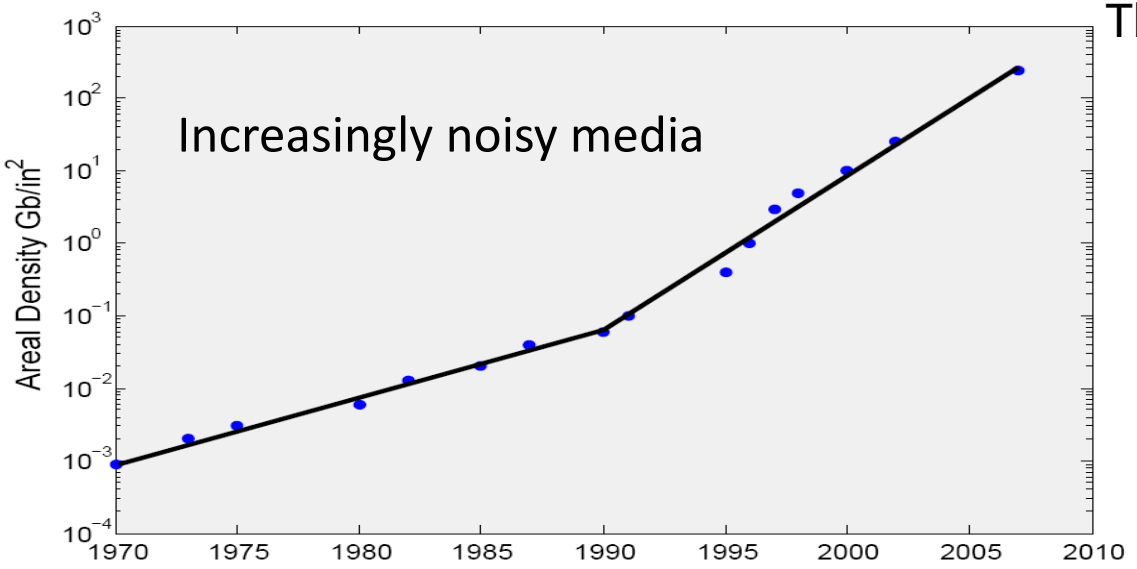
Introduction and Motivation

Error Rate Endurance Retention

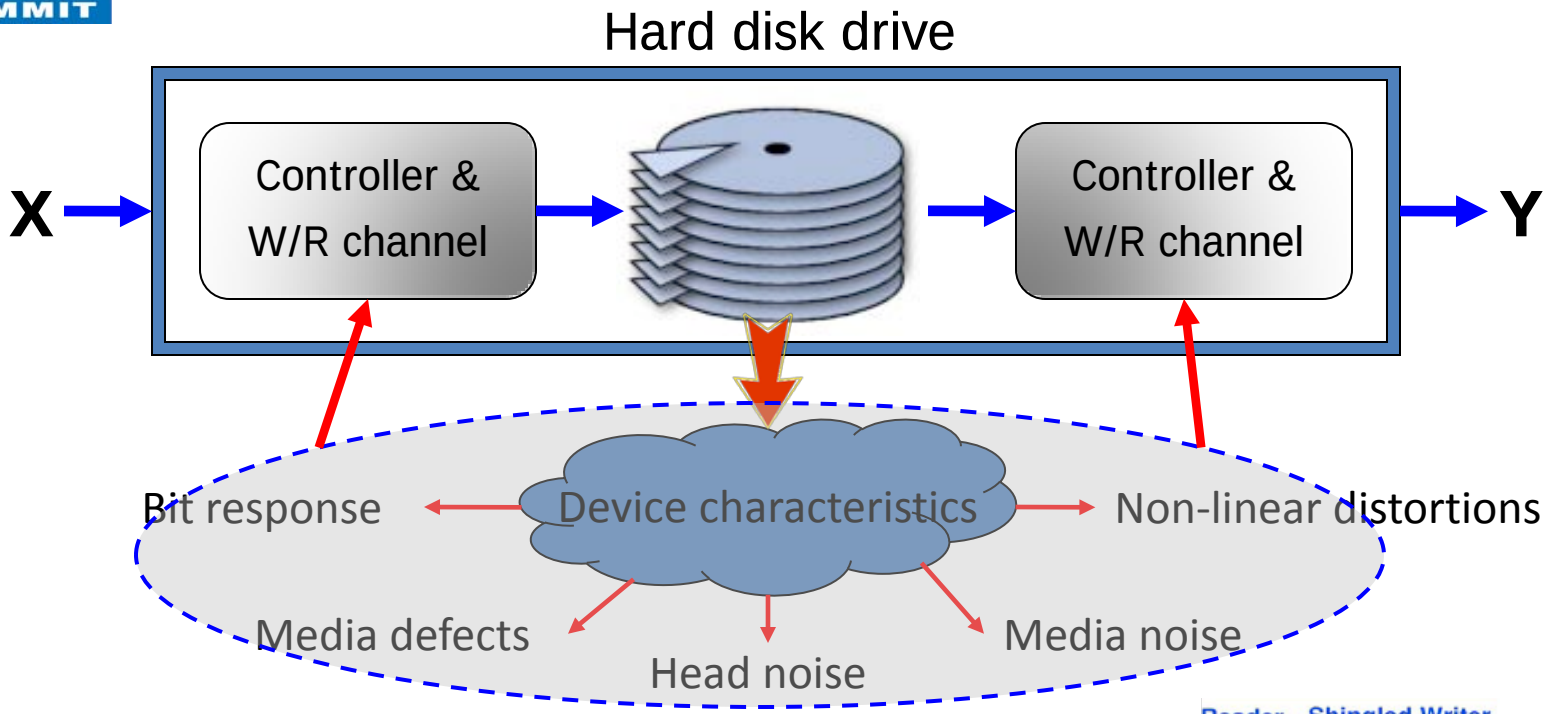




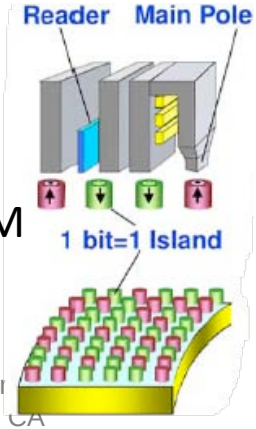
Hard disk drive



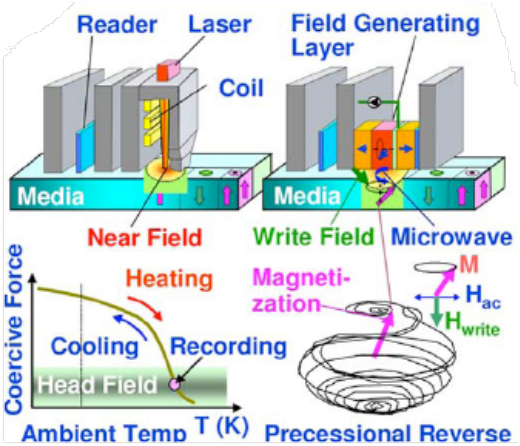
In-Depth Knowledge of Storage Media



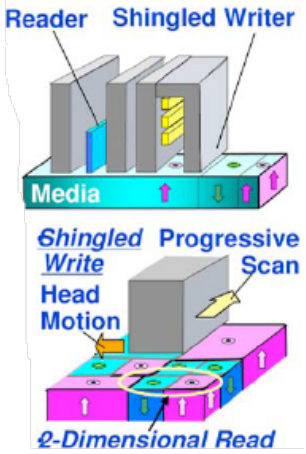
BPM



HAMR

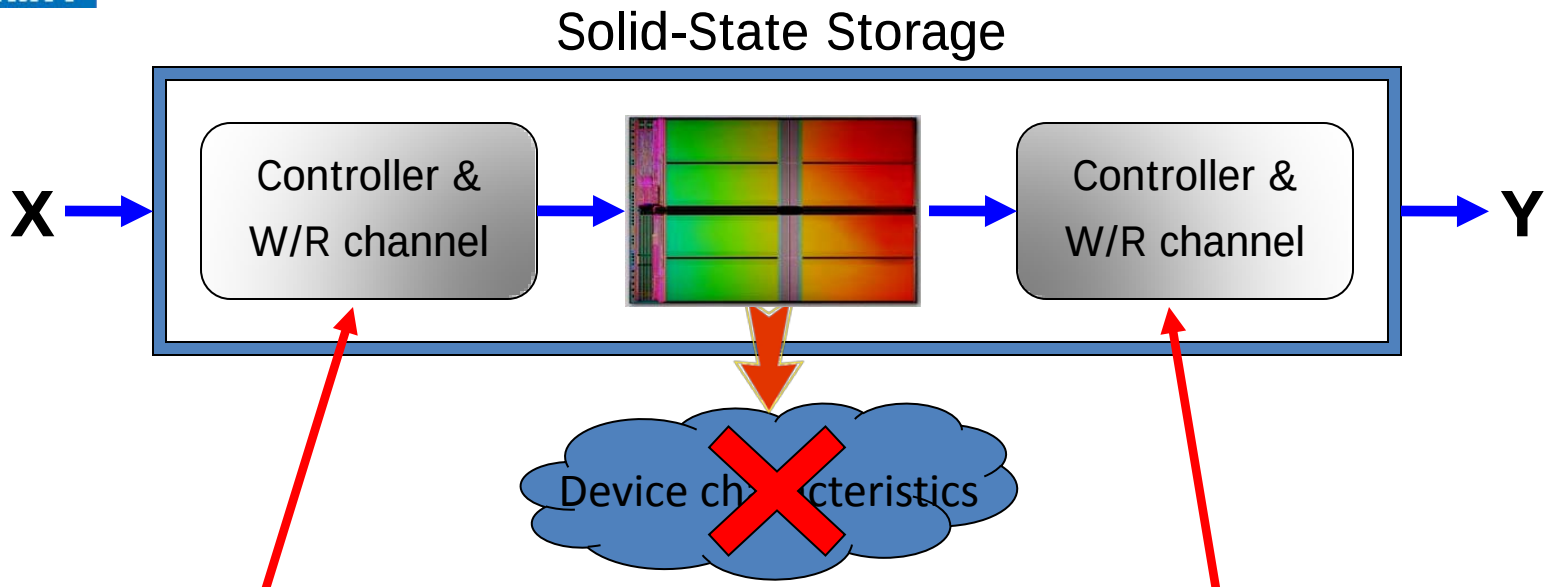


MAMR



TDMR

In-Depth Knowledge of Storage Media



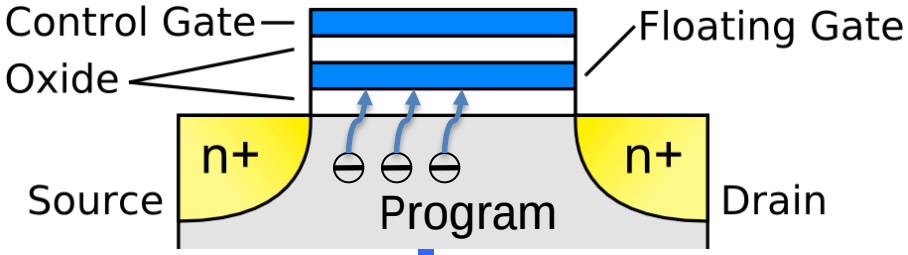
Write latency	Read latency	Erase latency	Endurance	Redundancy	…

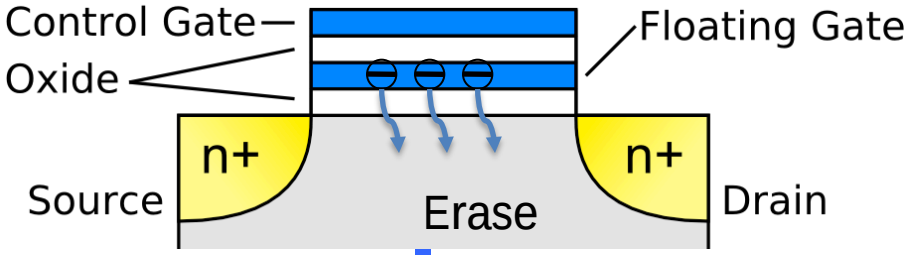
Painless system designers' life

Lost system optimization opportunities

Device-Aware SSD System Design

Device Wear-Out



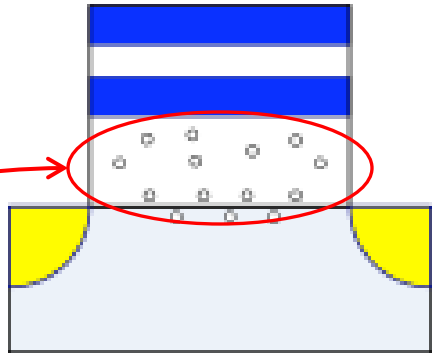


Modulate transistor threshold voltage

Program/erase (P/E) cycling

↓

Accumulates charge traps

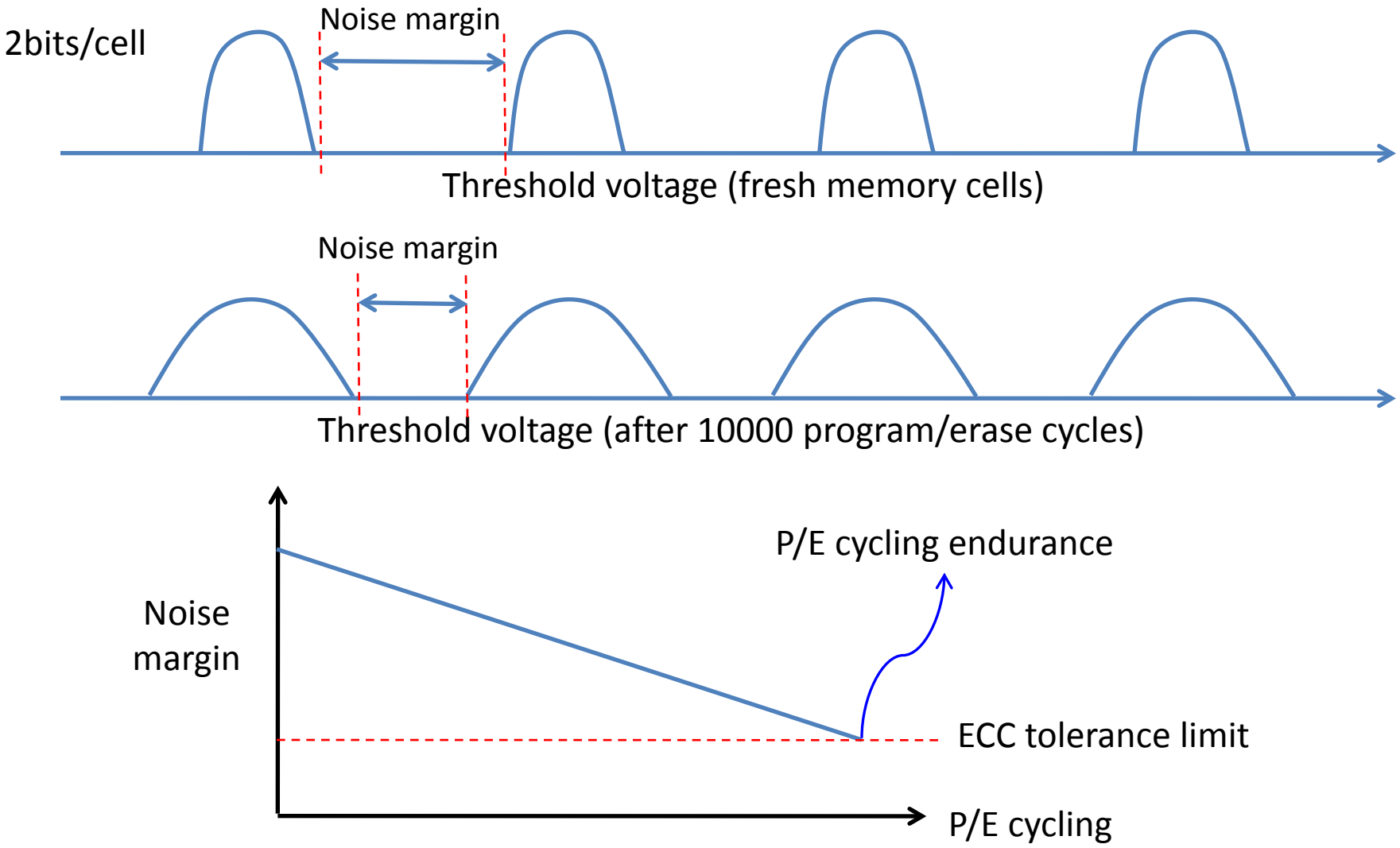


↓

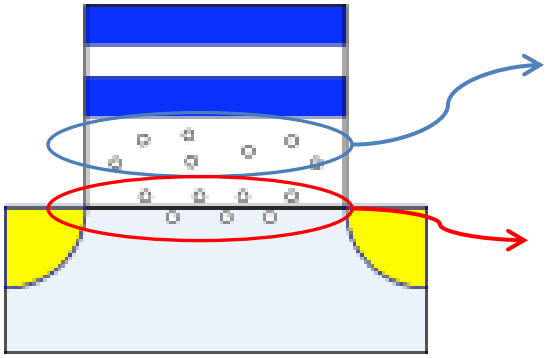
1. Larger random telegraph noise
2. Larger threshold voltage degradation during retention
3. Larger program/read disturb

→ Smaller noise margin

Device Wear-Out



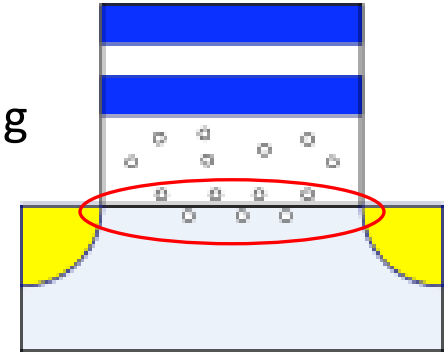
Device Wear-Out Recovery



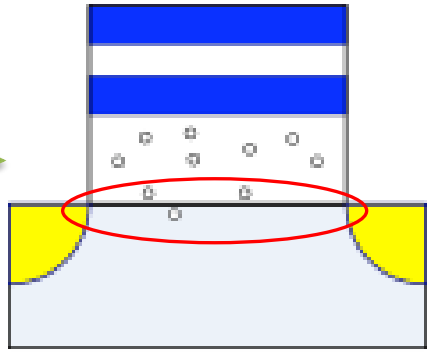
Oxide trap

Interface state trap → Recover over time

P/E cycling



Time

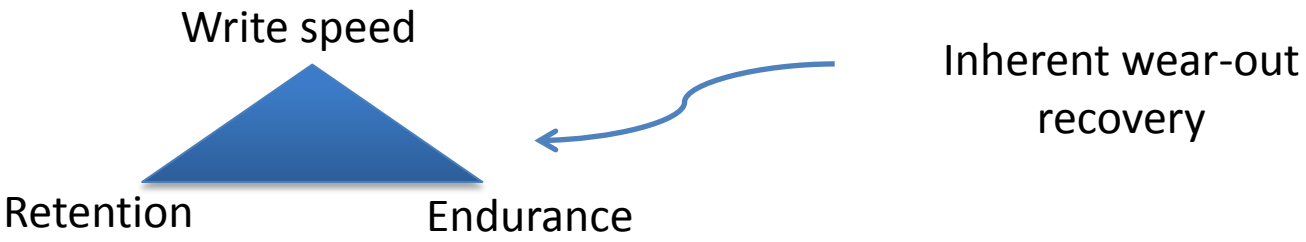


Noise margin ↑

Temperature ↑ → Recovery speed ↑

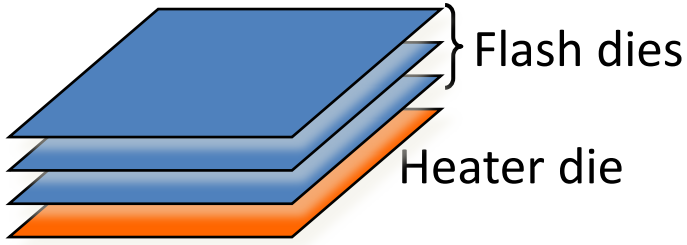
Self-Healing SSD?

- Explicitly leverage this device wear-out recovery phenomenon in FTL
  - Re-think of how to utilize existing over-provisioning?
  - Keep track of the history of environment temperature
- Intentionally operate SSDs under higher environment temperature

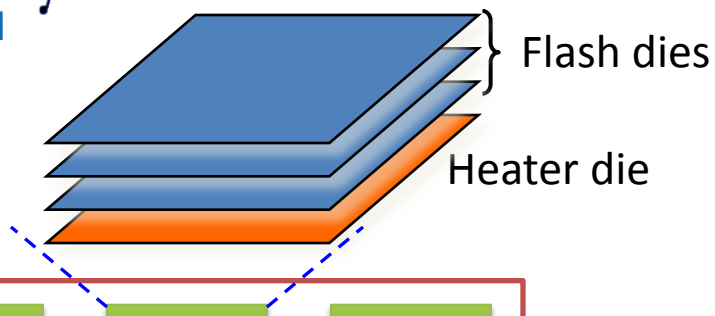


Most Aggressive Scenario

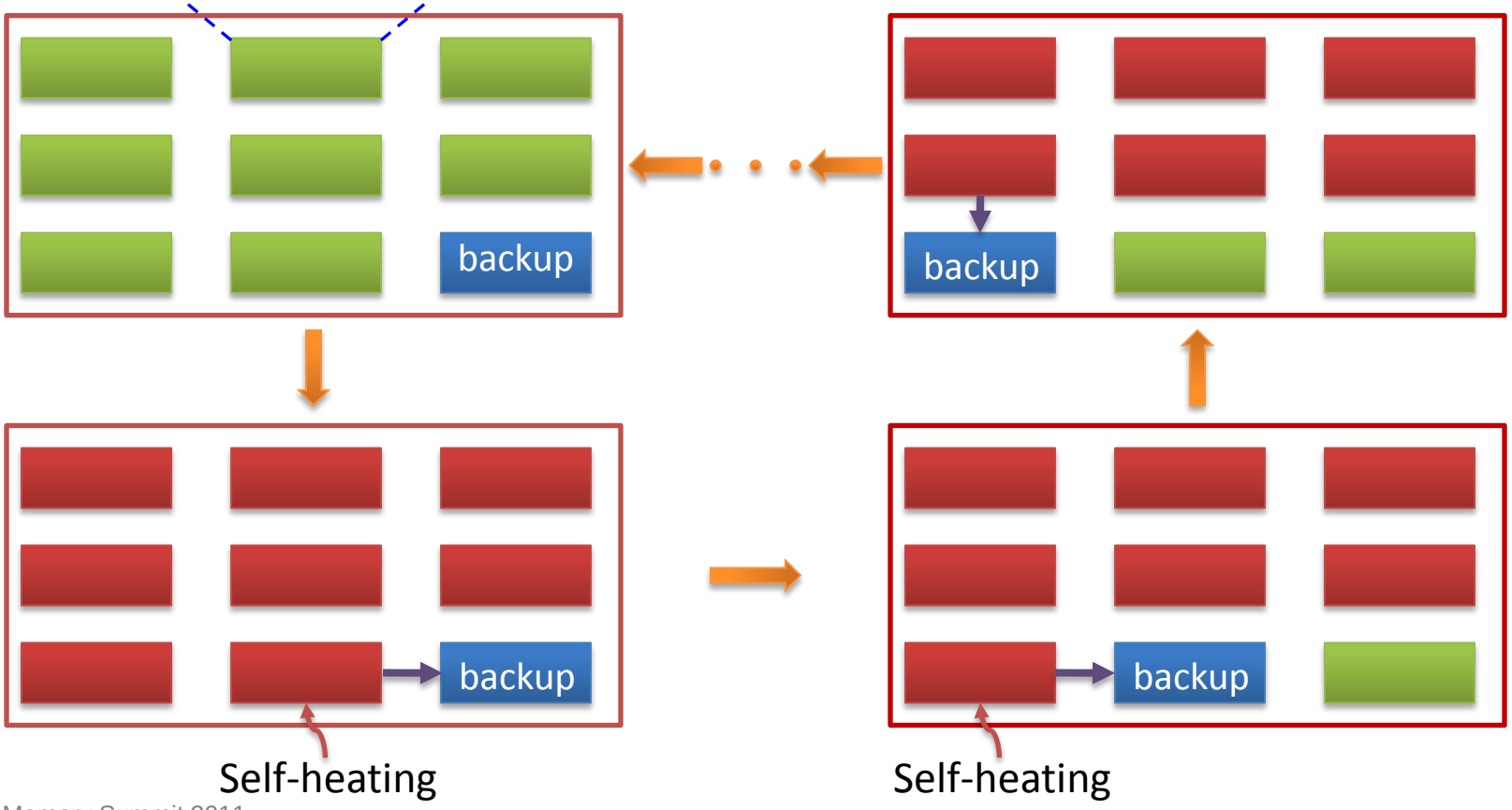
- 3D-enabled self-heating flash chip



Self-Healing SSD?

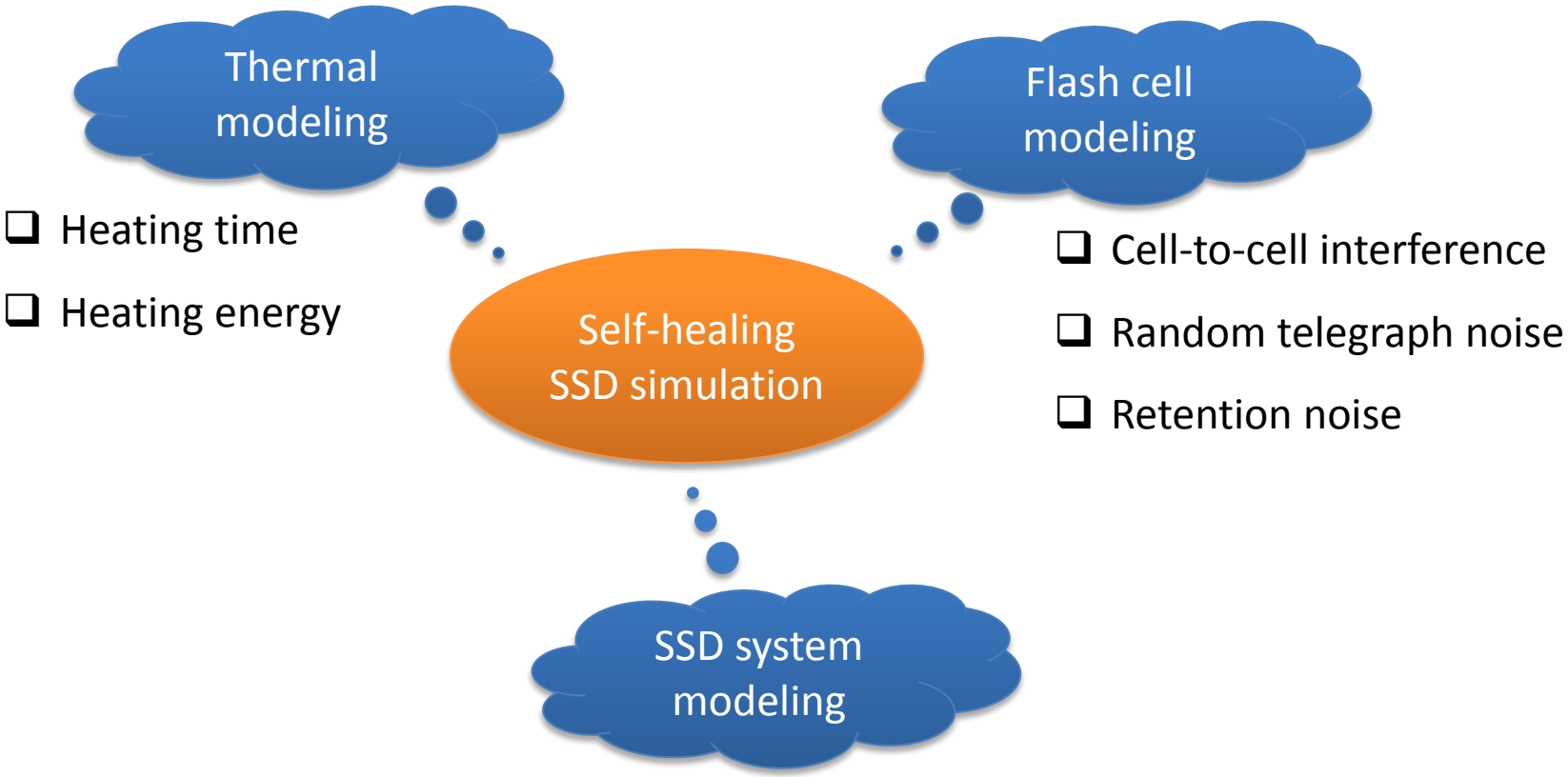


- ✓ Improve P/E cycling endurance
- ✓ Improve write speed
- ✓ Increase retention time



A Preliminary Evaluation

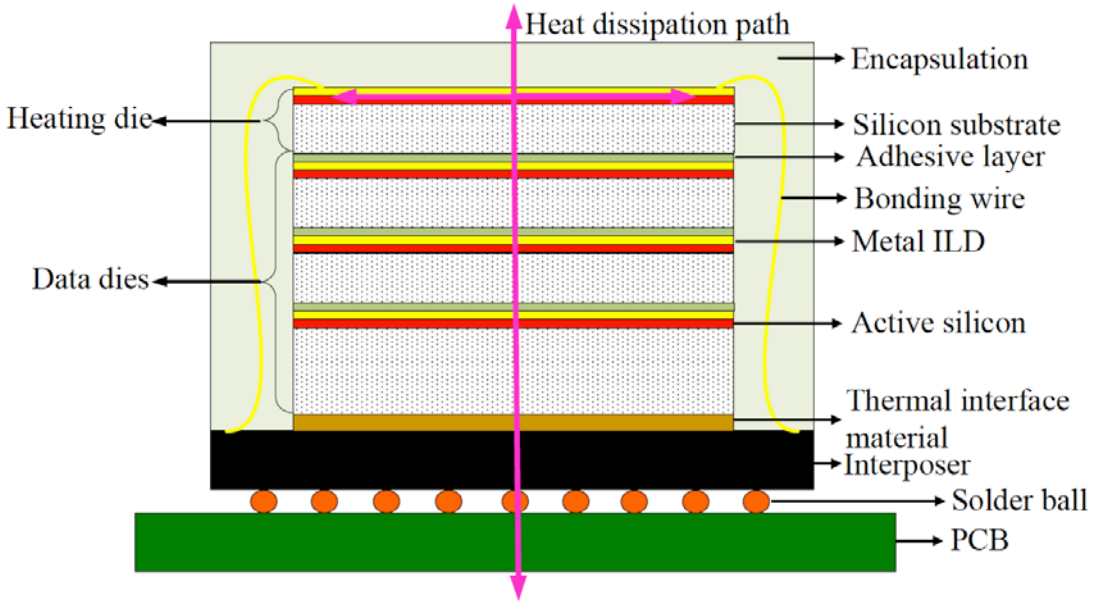
Improve P/E cycling endurance



Thermal Simulation Setup

HotSpot thermal modeling

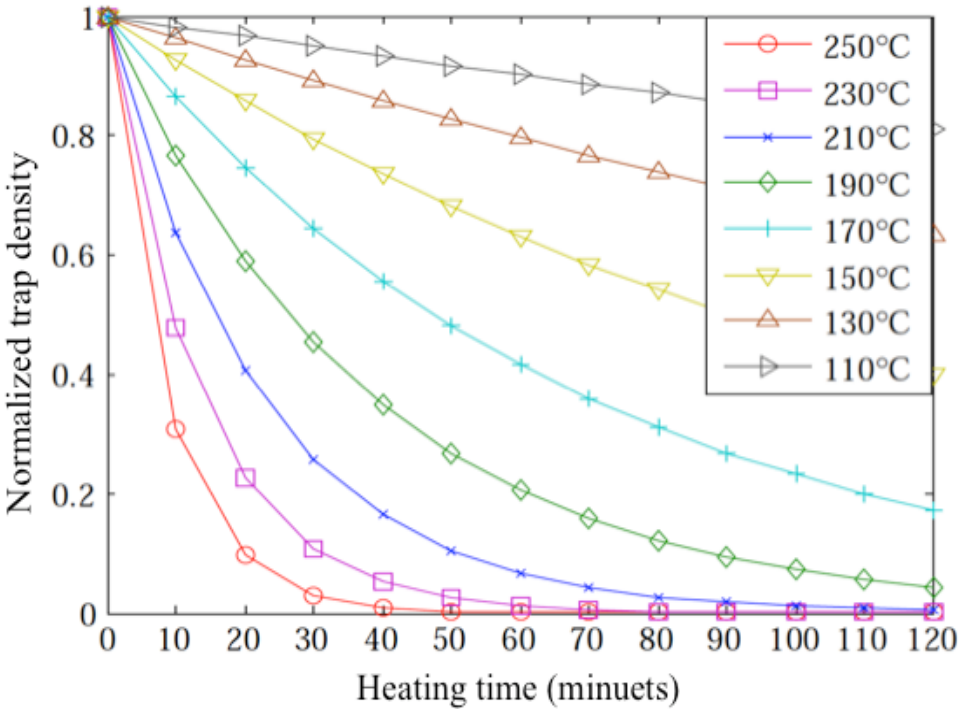
3D chip structure

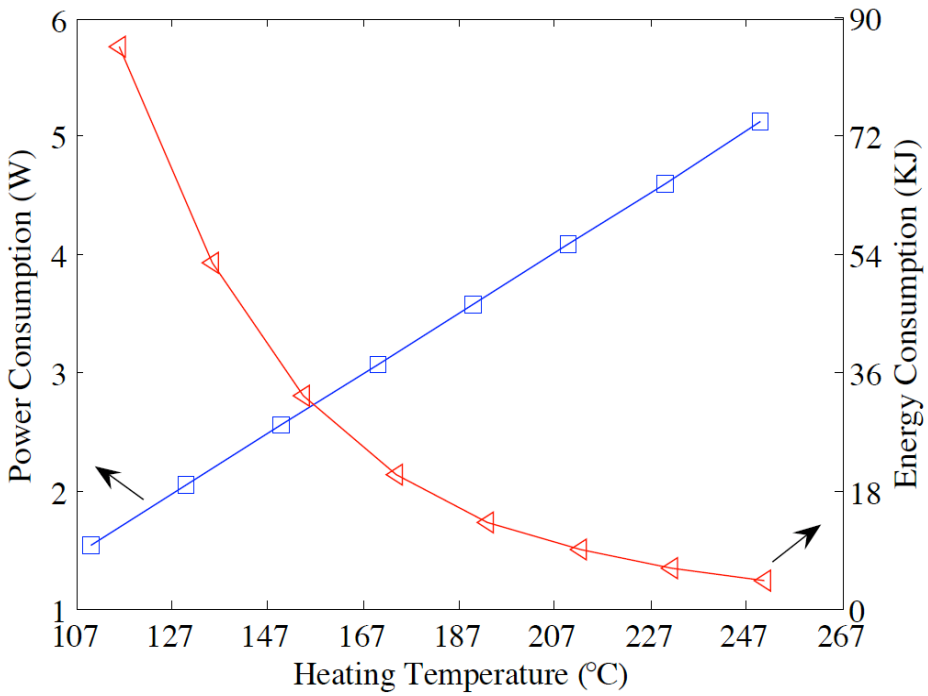


Layer	Thermal conductivity	Thickness
Encapsulation	0.453 W/(mk)	1.0 mm
Silicon substrate	100 W/(mK)	50 μm
Adhesive material	4 W/(mk)	4 μm
Metal ILD	200 W/(mk)	8 μm
Active silicon	100 W/(mk)	2 μm
Thermal interface material	4 W/(mk)	20 μm
Interposer	2 W/(mk)	0.4 mm
Solder balls	16.7 W/(mk)	0.94 mm
PCB	3 W/(mk)	2 mm

3D chip setup

Thermal Simulation Results

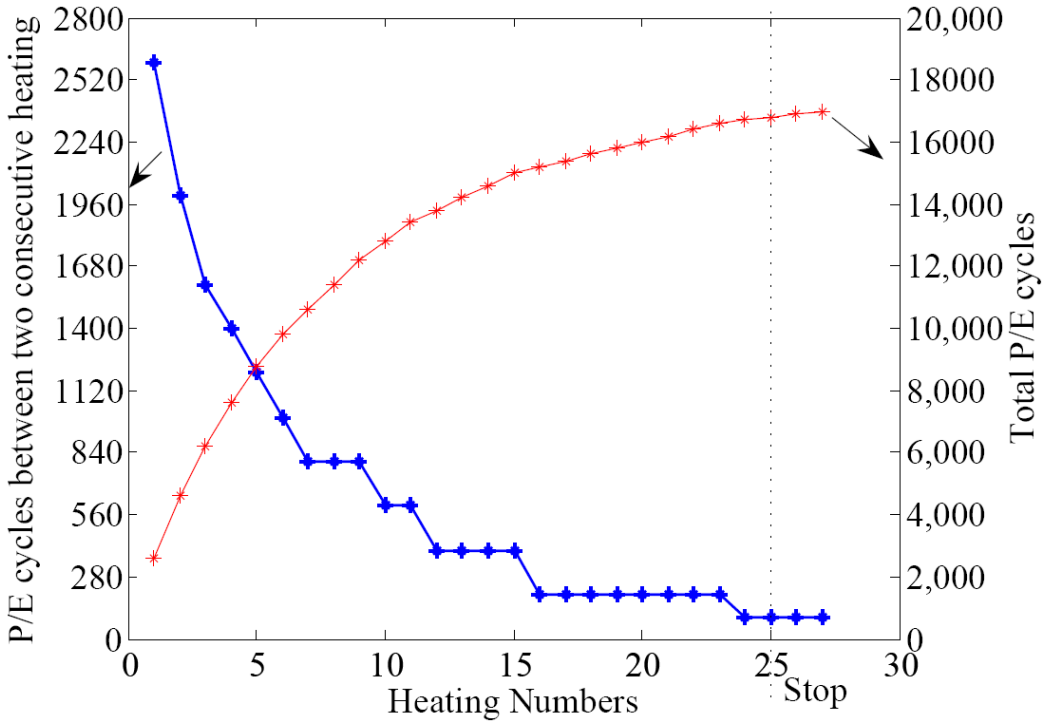




- 1.5W to 5.1W power consumption when temperature changes from 110 to 250C
- Choose 200°C as the target heating temperature
- ~35 minutes for 80% interface state traps to recover

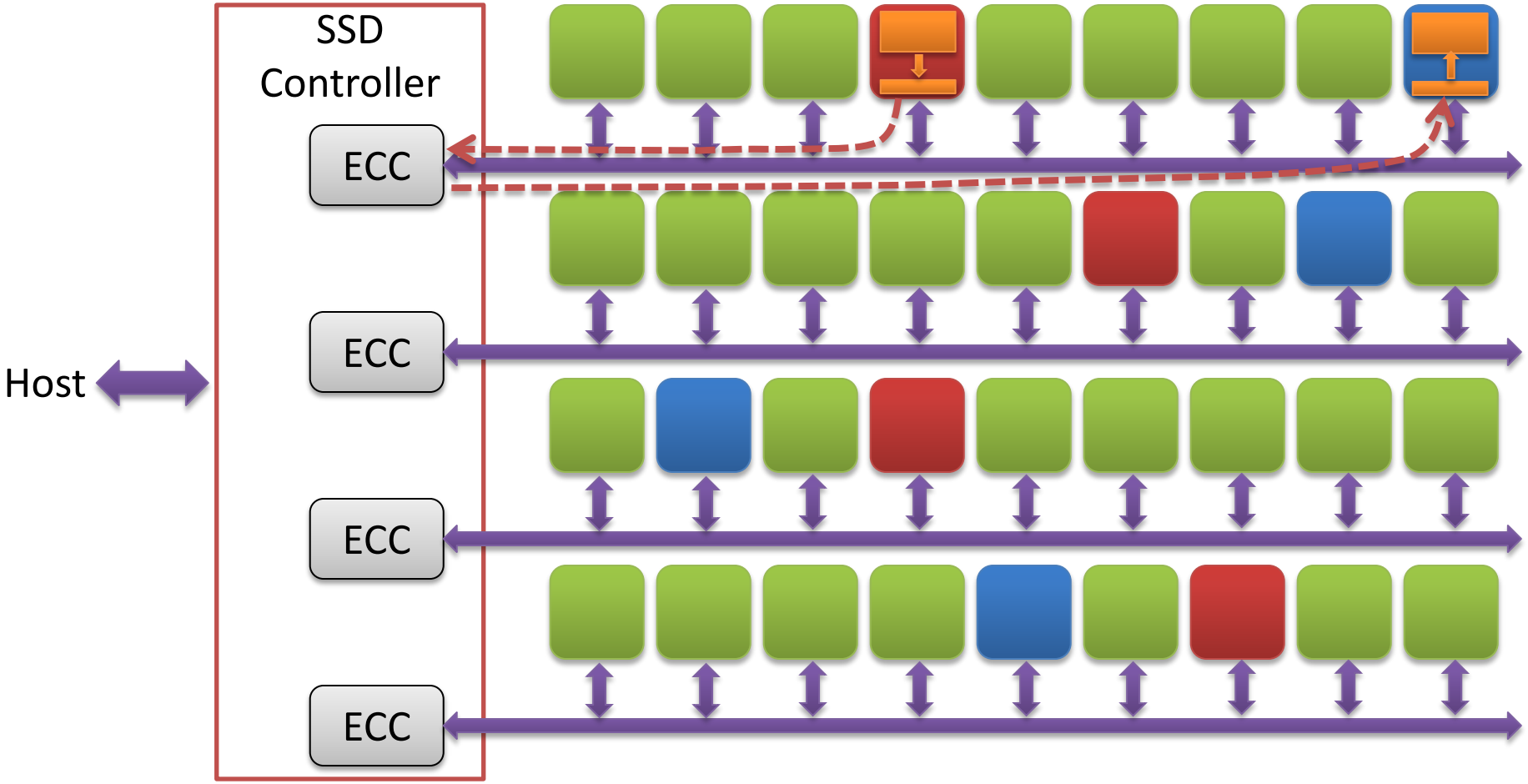
P/E Cycling Endurance Improvement

- Allowable worst-case memory read raw BER of 2.04e-3
- 10-year retention limit
- Include trap recovery under normal temperature (45° C)
- Self-healing trigger BER 1.50e-3
- Three times longer cooling time

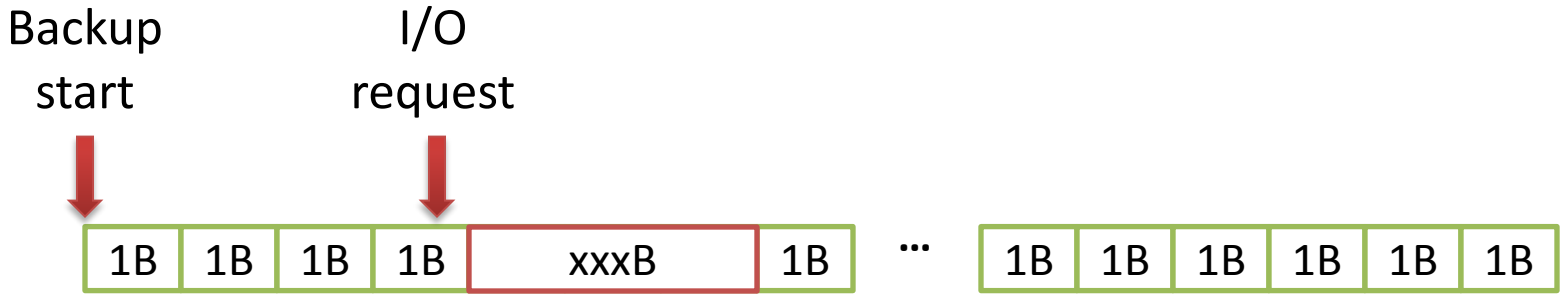


P/E cycling endurance: 3000 → 17400

Data Backup During Chip Self-Heating



Data Backup During Chip Self-Heating

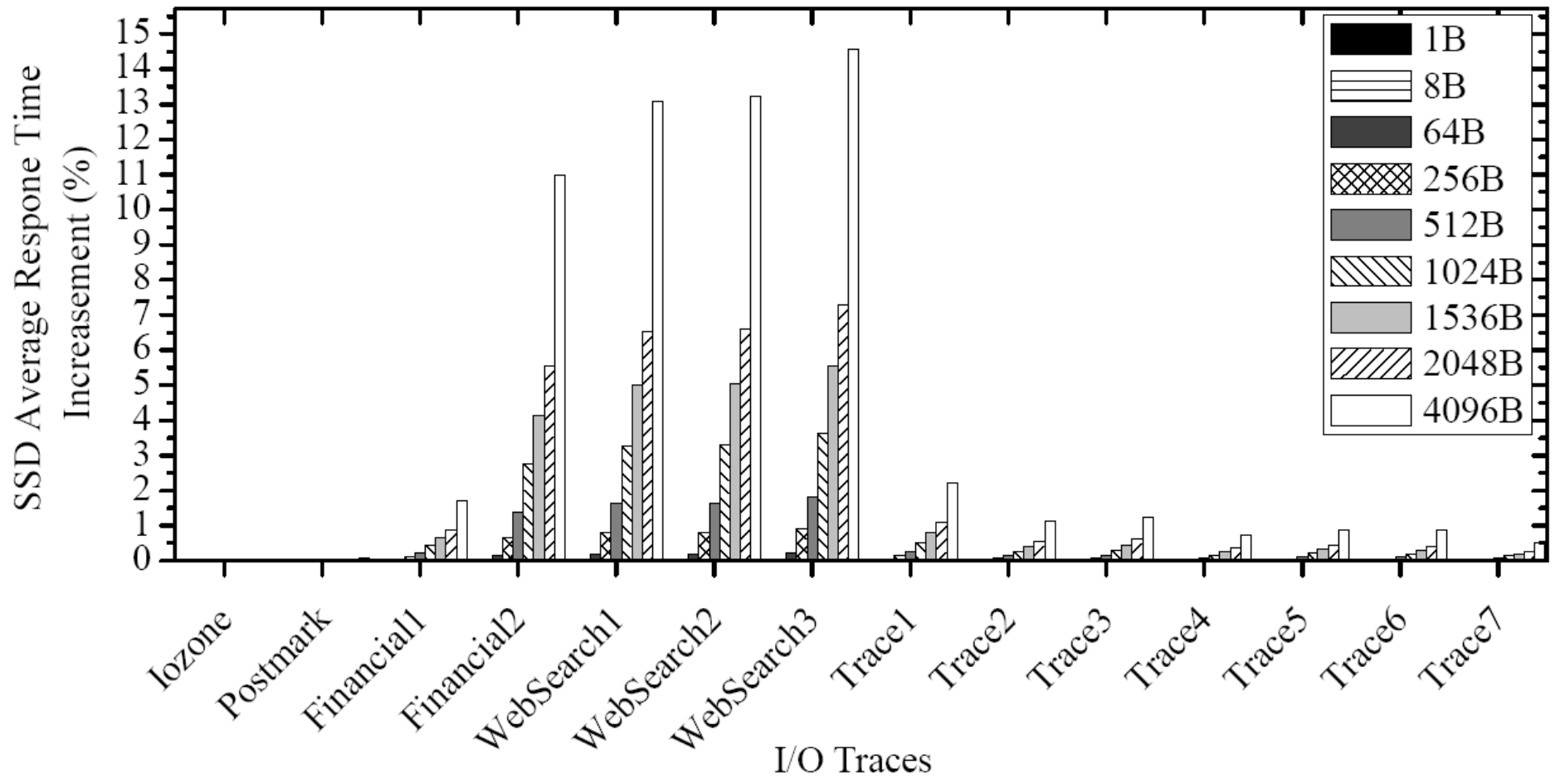


- Initialize data backup when SSD is idle
- I/O request can interrupt data backup operation
- Impact of data backup granularity on I/O request response time
- Need to handle read and write conflict

Simulation Setup

- DiskSim simulator with SSD model patch
- 2 dies/chip, 8-bit I/O bus and a number of common control bus
- 2 planes/die, 2048 blocks/plane, 64 pages/block, 8 sectors/page, 512bytes/sector
- 2 channels, 17 flash chips/channel including one backup chip
- Backup chip on each channel only backups the data of the flash chips on the same channel
- ONFI 2.0, 133MB/s, read access time 50µs, program time 600µs

Simulation Setup



Summary

- Continuous technology scaling demands true device-aware SSD system design
- How to exploit memory cell wear-out recovery?
  - Explicitly leverage this wear-out recovery phenomenon in FTL
  - A more aggressive scenario: self-heating NAND flash memory chips
    - SSD controller scheduling and data backup strategy
    - Simulation based on detailed thermal, flash memory cell, and SSD system modeling
- Comprehensive cross-layer optimization: **an open question**