Aegis: Partitioning Data Block for Efficient Recovery of Stuck-at-Faults in Phase Change Memory

Jie Fan, Jiwu Shu, Youhui Zhang, and Weimin Zhen

Song Jiang

Tsinghua University

Wayne State University
Stuck-at Faults in PCM

- PCM has limited endurance.

- Stuck-at faults occur when memory cell fails to change its value.
  - It is a major type of faults in PCM.
  - This type of faults is permanent and accumulates.
  - Values in such faulty cells can still be read.

- Inversion-based correction
  - Partition data block into a number of groups and exploit the fact that stuck-at values are still readable (e.g., SAFER).
  - Each group can tolerate only one fault.

- Proposal of an efficient partition scheme separating faults into different groups.
PCM bits are laid out on an $A \times B$ Cartesian plane.

Aegis considers all points on a line as a group.

**Any two bits in the same line will not be in the same line** after Aegis changes slope of the lines.

Aegis distributes faults more evenly to tolerate more faults with lower overhead.
Fine-grained Heterogeneity

Traditional big.LITTLE Architecture

Big

$\rightarrow$

Little

$\rightarrow$

Memory

Transfer Overhead: $\sim 20K$ cycles
Fine-grained Heterogeneity

Traditional big.LITTLE Architecture

Transfer Overhead: ~20K cycles
Fine-grained Heterogeneity

Traditional big.LITTLE Architecture

- Big
- Little
- Memory

Transfer Overhead: ~20K cycles

Composite Cores Architecture

- Big Backend
- Little Backend
- Shared Frontend
- Memory

Share Architectural State

Transfer Overhead: ~35 cycles
Traditional Reactive Controllers

![Graph showing the percentage of time spent on little instructions per quantum for Oracle and Reactive controllers. The graph plots the number of instructions per quantum on the x-axis and the percentage of time spent on little instructions on the y-axis. The Oracle controller shows a significant decrease in the percentage of time spent on little instructions as the number of instructions per quantum increases, while the Reactive controller shows a more consistent percentage.](image-url)
Don’t React – Predict!

Traditional Reactive Controllers

% Time Spent On Little

# Instructions per Quantum

Oracle
Reactive
Traditional Reactive Controllers

Don’t React – Predict!

Code repeats (loops, functions)

Behavior repeats in the same program context
Don’t React – Predict!

Code repeats (loops, functions)

Behavior repeats in the same program context

Traditional Reactive Controllers

- Code repeats (loops, functions)
- Behavior repeats in the same program context

Online Trace-Based Predictive Controller

- Construct Super-Trace
- Predict next Super-Trace
- Predict next backend

Feedback