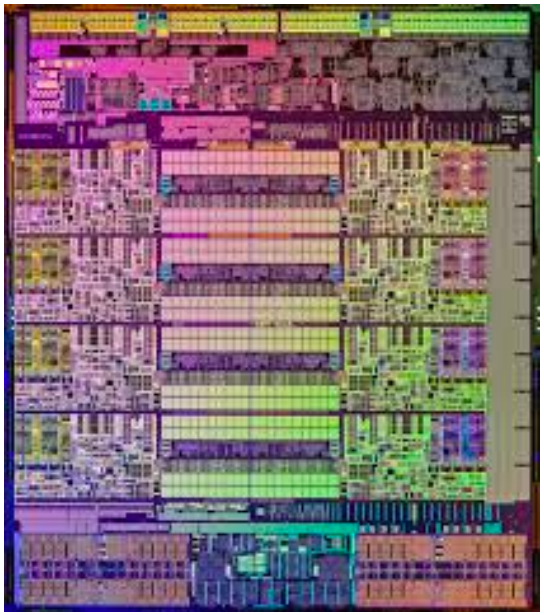
SUDOKU: TOLERATING HIGH-RATE OF TRANSIENT FAILURES FOR ENABLING SCALABLE STTRAM

Prashant J. Nair | The University of British Columbia

Bahar Asgari
Moinuddin K. Qureshi | Georgia Institute of Technology

INTRODUCTION

Caches are key for enabling high performance processors



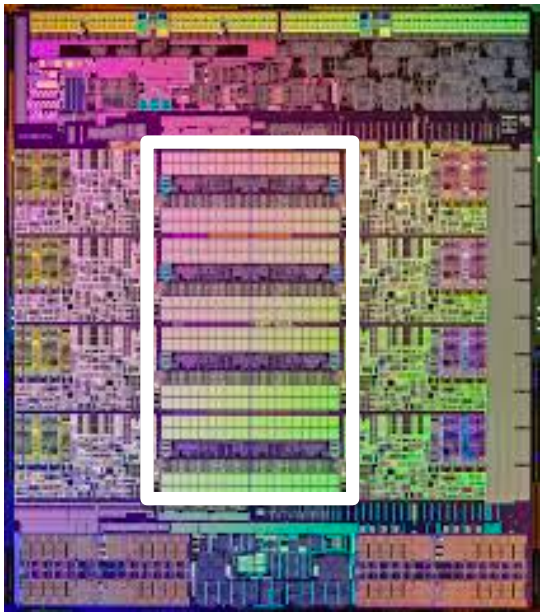
*Processor: Die-Shot**

*Source: Intel Haswell-E Die

INTRODUCTION

Last-Level Caches (LLC) → Tend to occupy the largest area

LLC → Static RAM (SRAM)

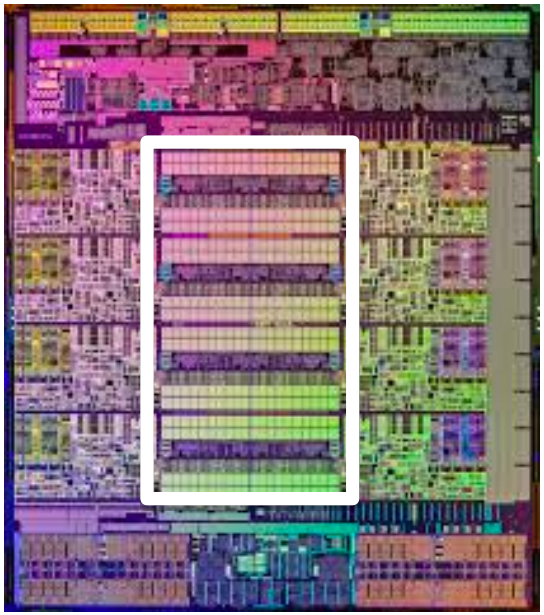


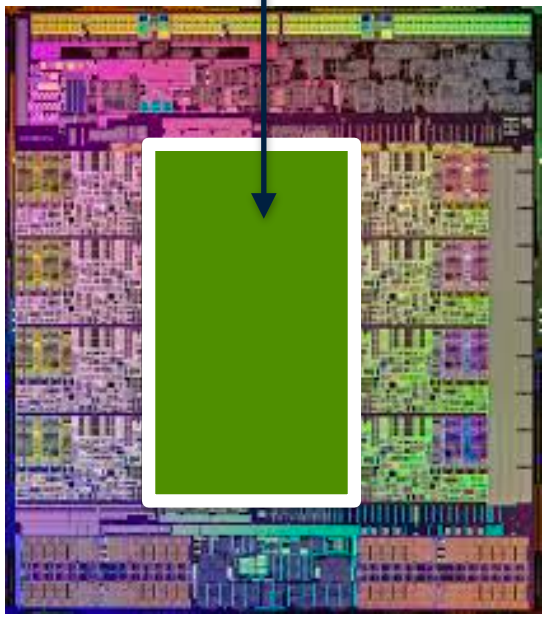
*Processor: Die-Shot**

*Source: Intel Haswell-E Die

INTRODUCTION

LLC → ~~Static RAM~~ → Spin Torque Transfer RAM





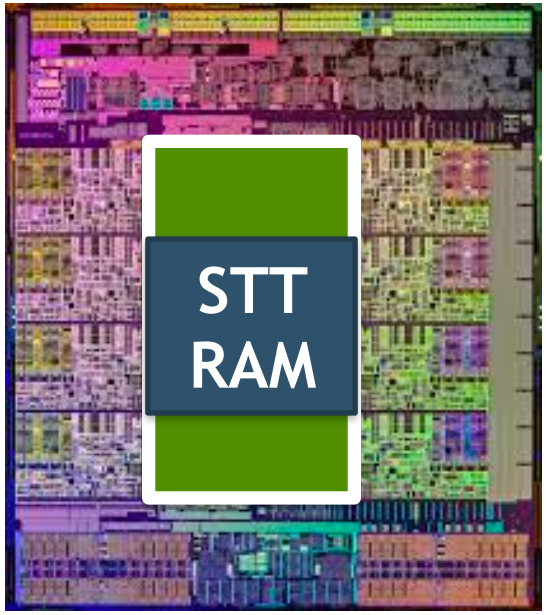
INTRODUCTION

Spin Torque Transfer RAM (STTRAM)

INTRODUCTION

Scaling STTRAM faces reliability concerns

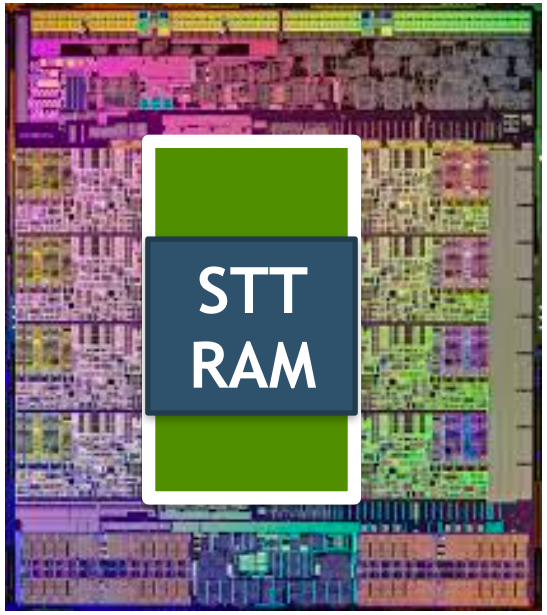
	SRAM	STTRAM
Capacity	✖	✔
Reliability	✔	✖



INTRODUCTION

Ideally we need to scale STTRAM with low overheads

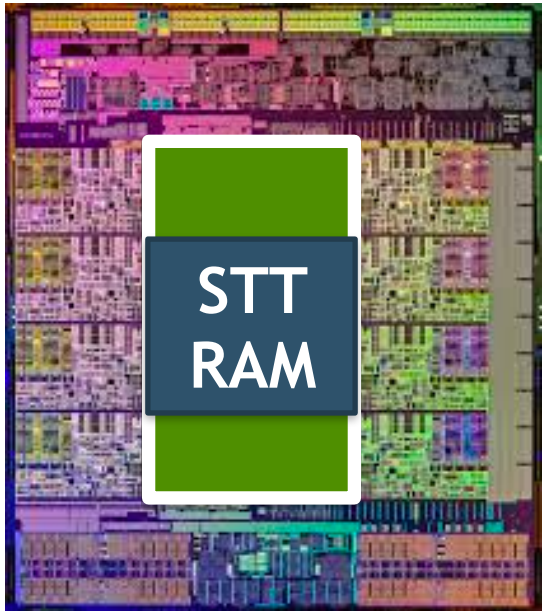
	Weak ECC	Strong ECC
Performance	✔	✖
Area Overhead	↓	↑
Reliability	✖	✔



INTRODUCTION

Enable STTRAM as a practical alternative to SRAM

	Weak ECC	Strong ECC	Goal
Performance	✔	✖	✔
Area Overhead	↓	↑	↓
Reliability	✖	✔	✔

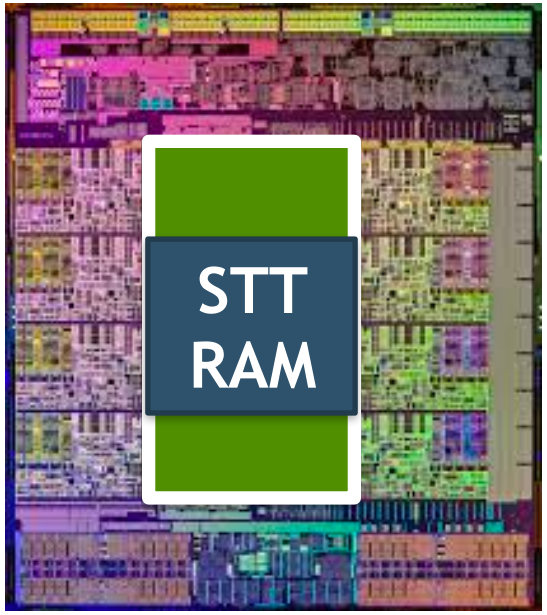


Goal: Reliable and Scalable STTRAM at Low Overheads

BACKGROUND

Average Retention Time = $1ns * e^{\Delta}$

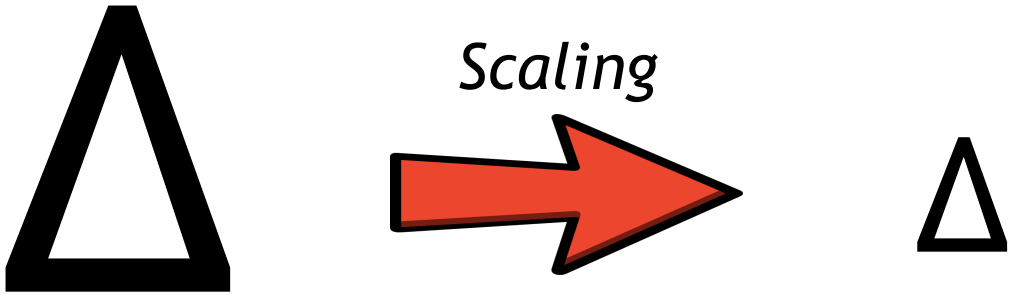
Δ = *Thermal Instability Factor*

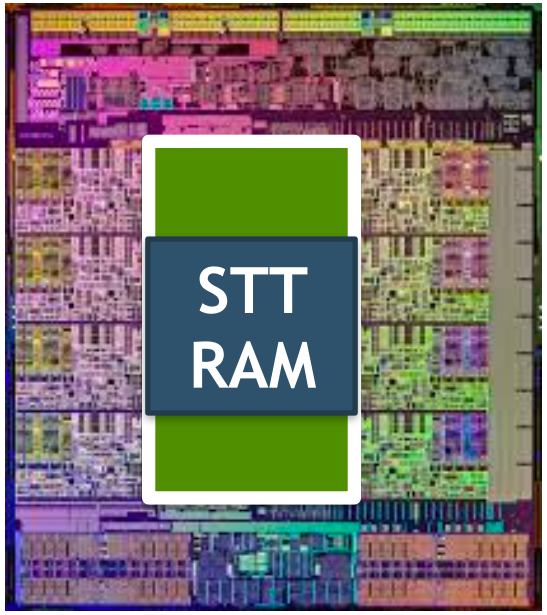


BACKGROUND

Average Retention Time = $1ns * e^{\Delta}$

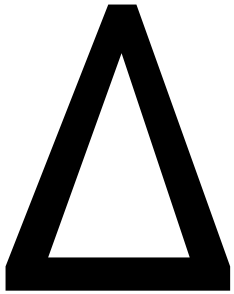
Δ = *Thermal Instability Factor*



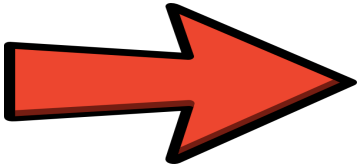


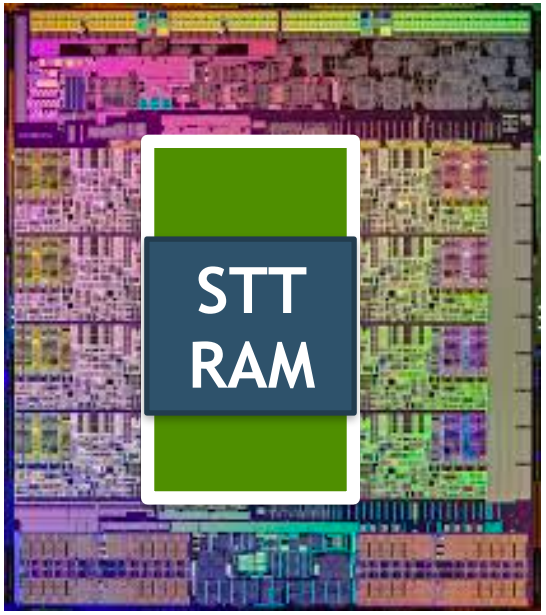
BACKGROUND

Polarization
of the cell



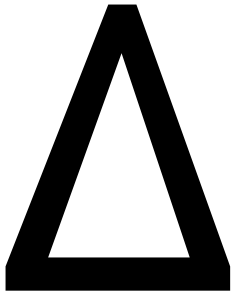
Scaling



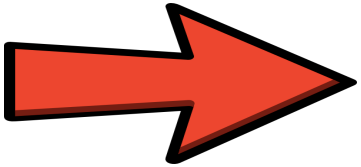


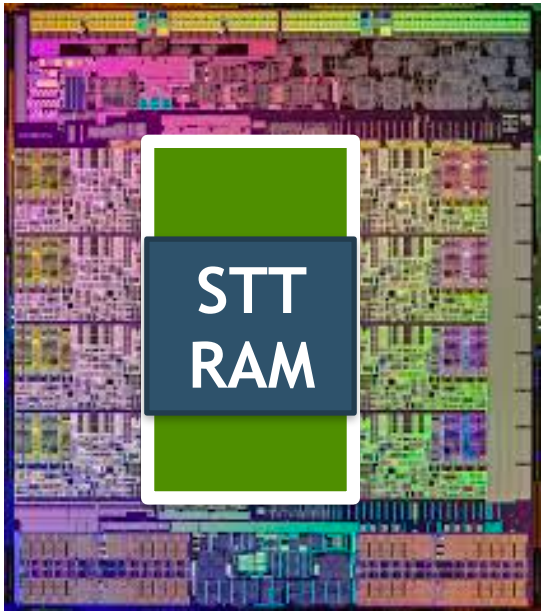
BACKGROUND

Polarization
of the cell



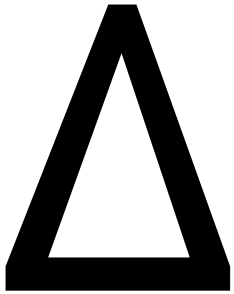
Scaling



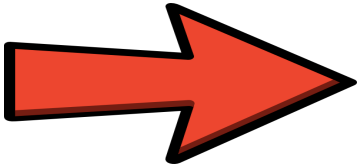


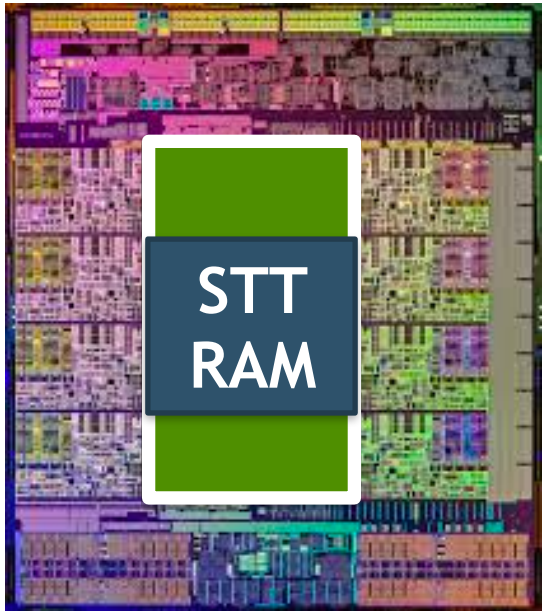
BACKGROUND

Polarization of the cell



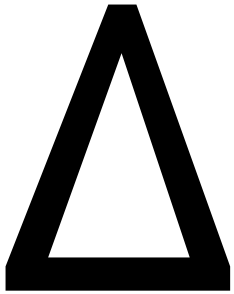
Scaling



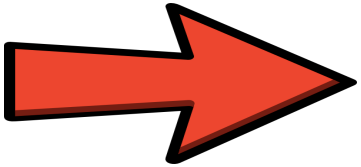


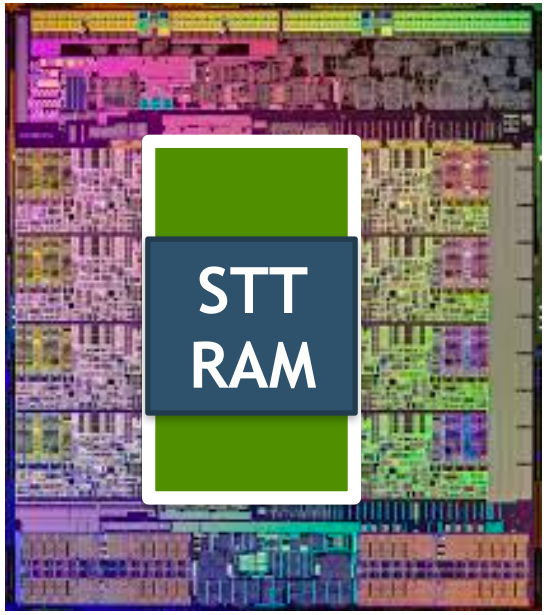
BACKGROUND

Polarization of the cell

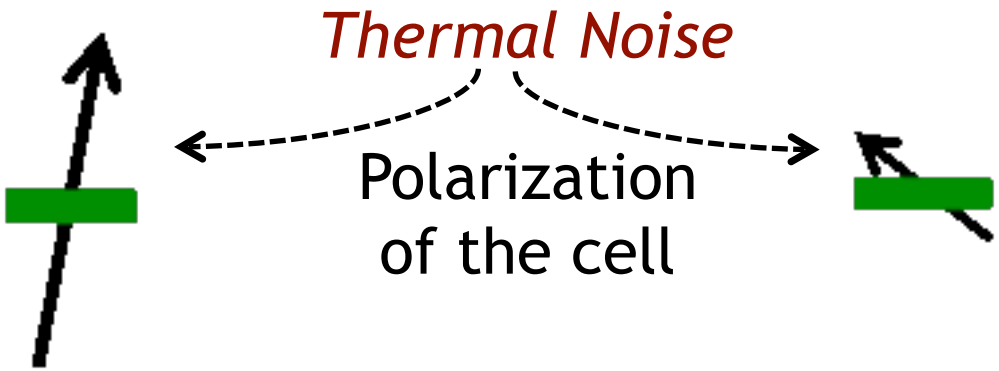


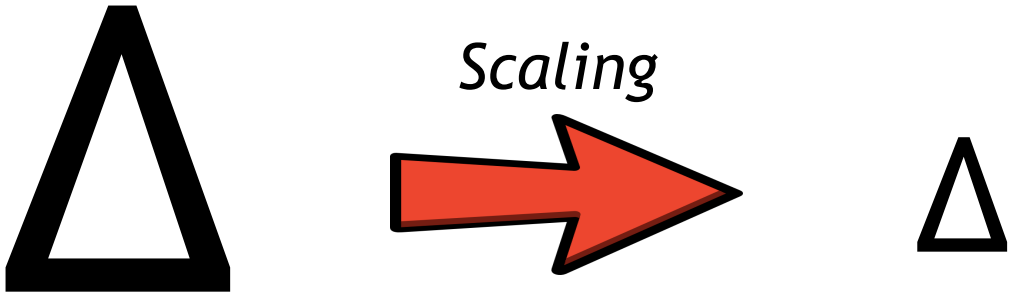
Scaling



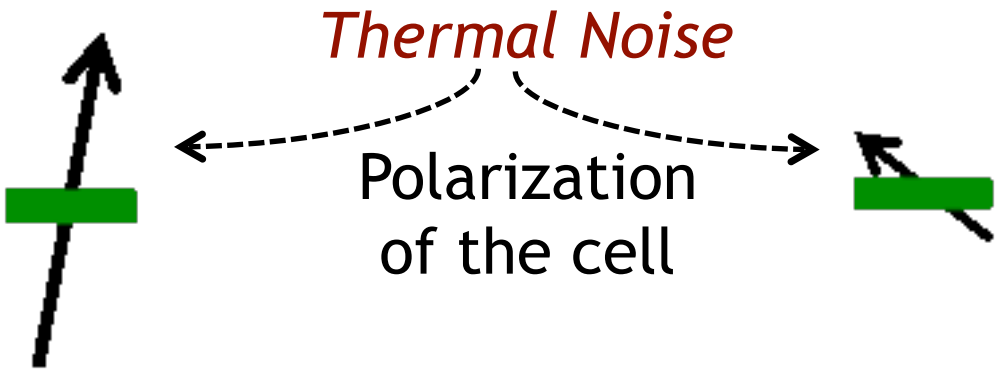


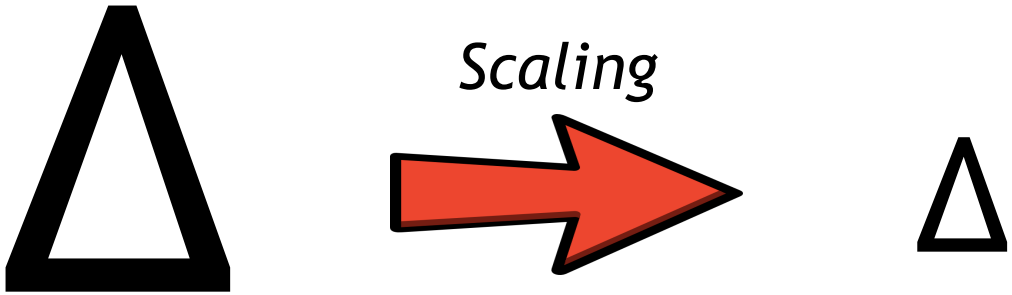
BACKGROUND



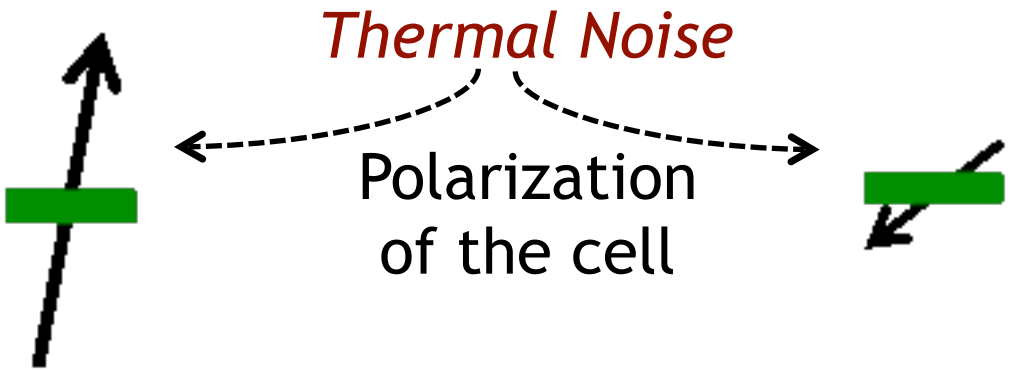


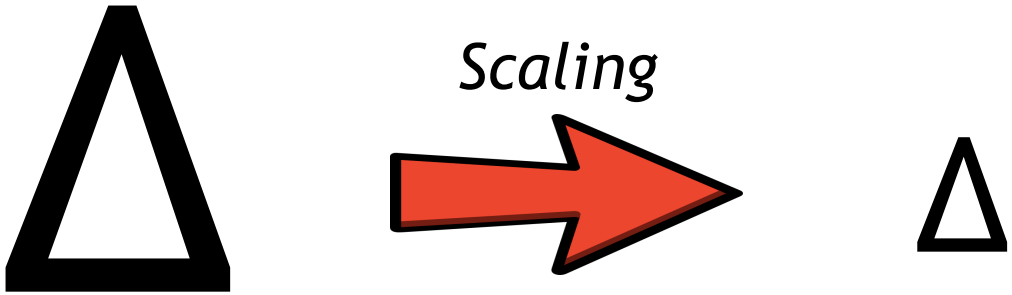
BACKGROUND

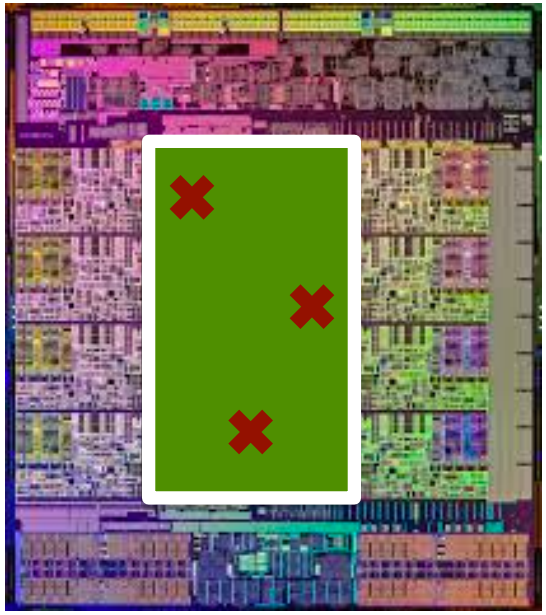




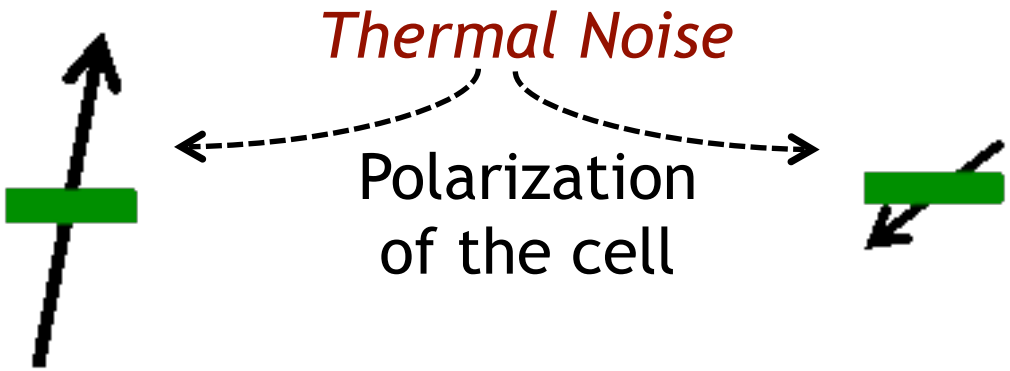
BACKGROUND

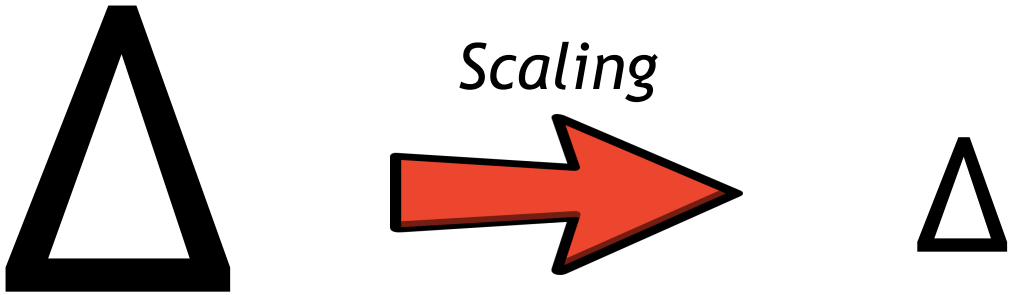


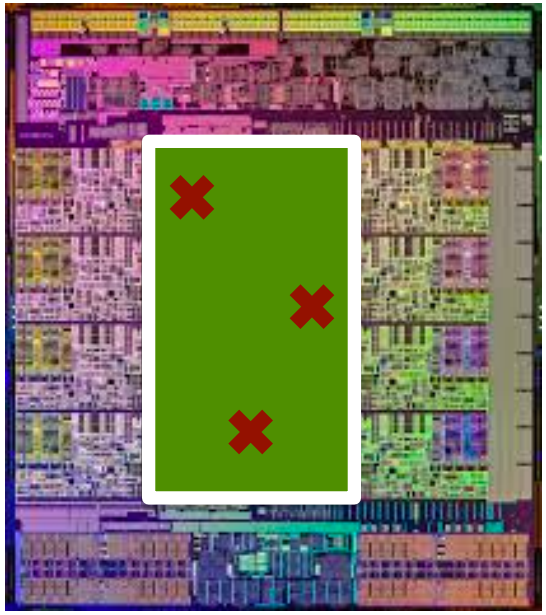




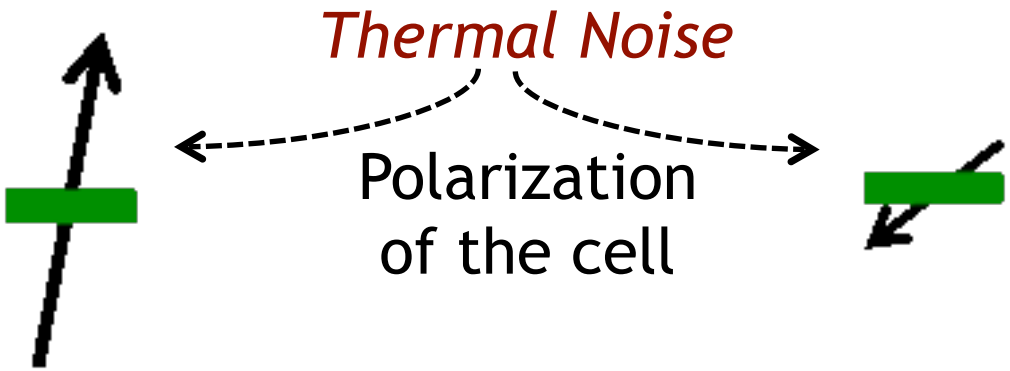
BACKGROUND

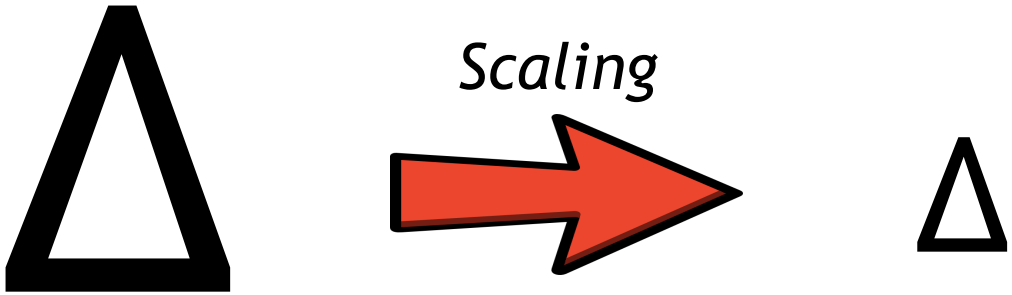


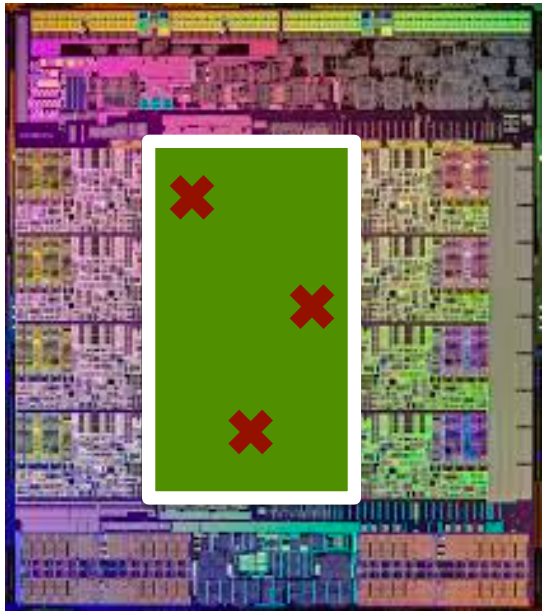




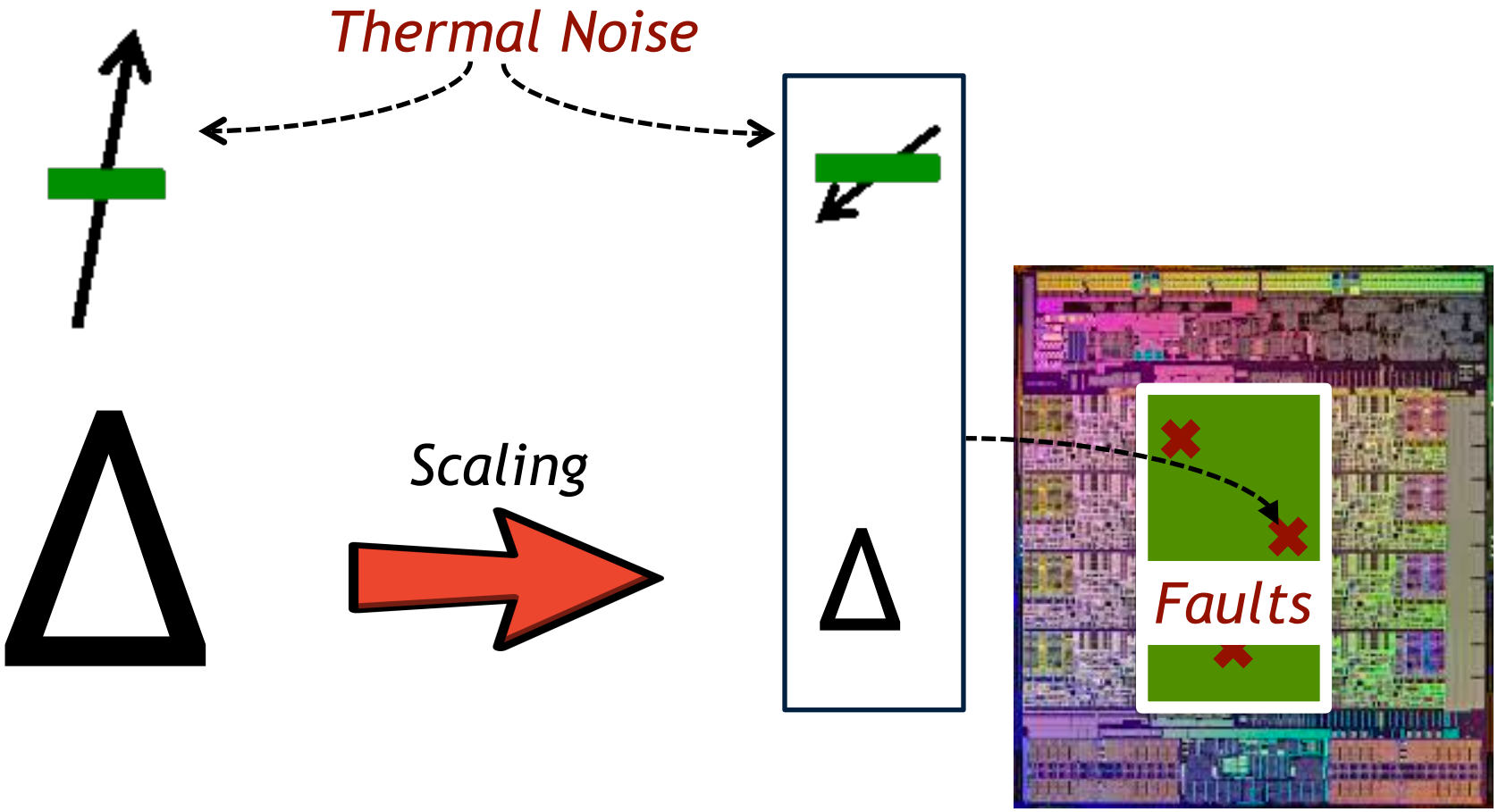
BACKGROUND



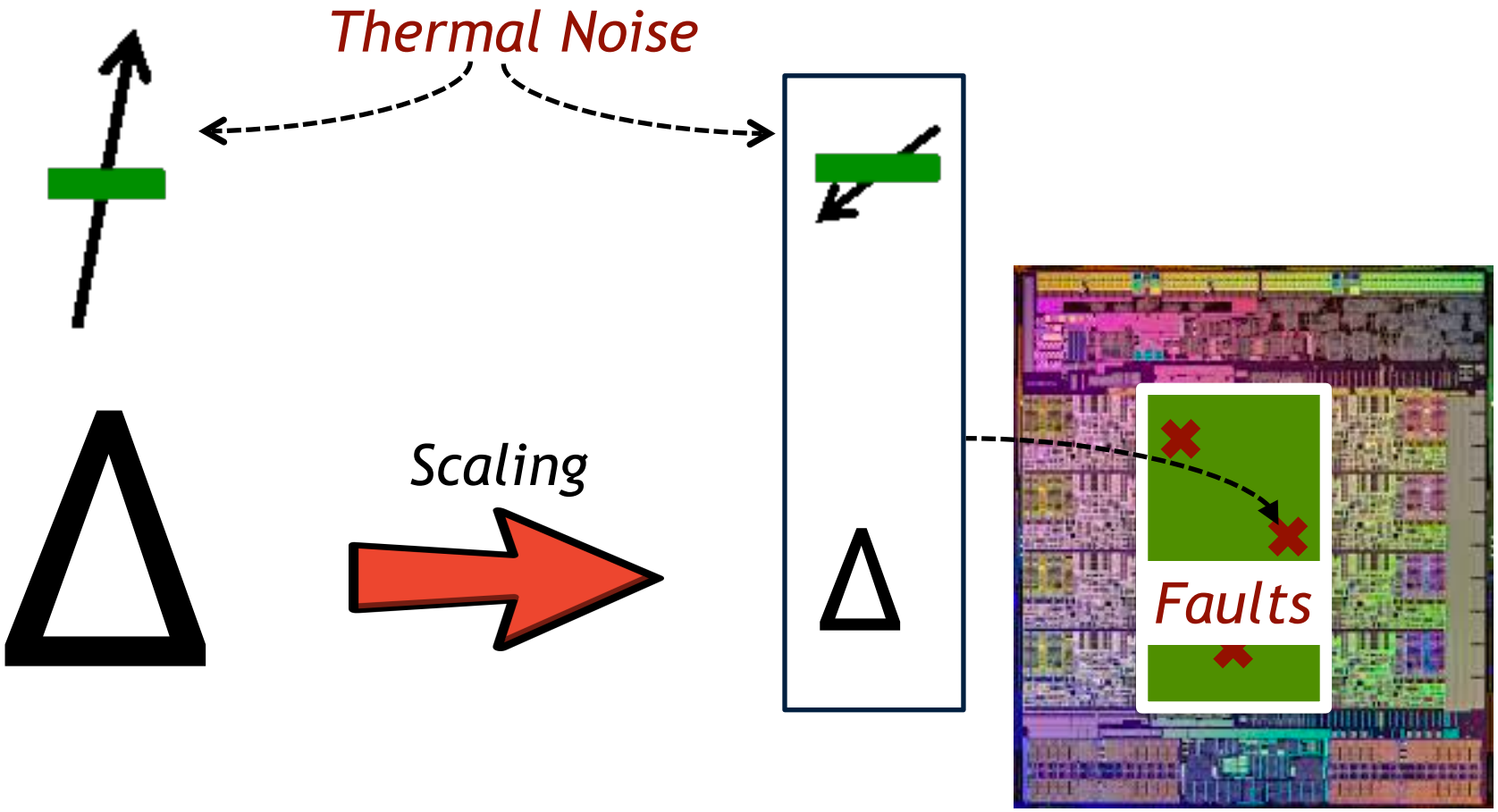




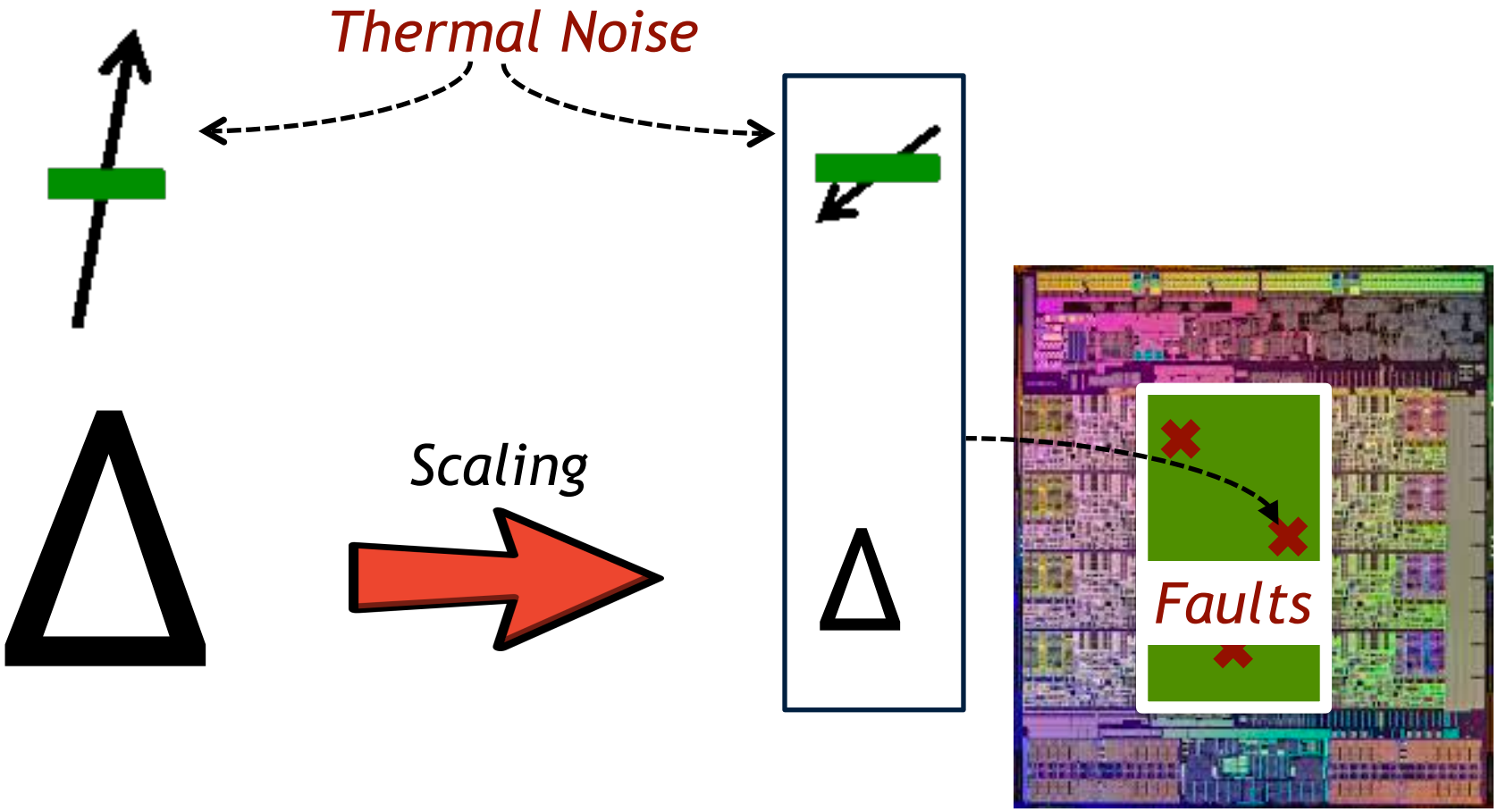
BACKGROUND



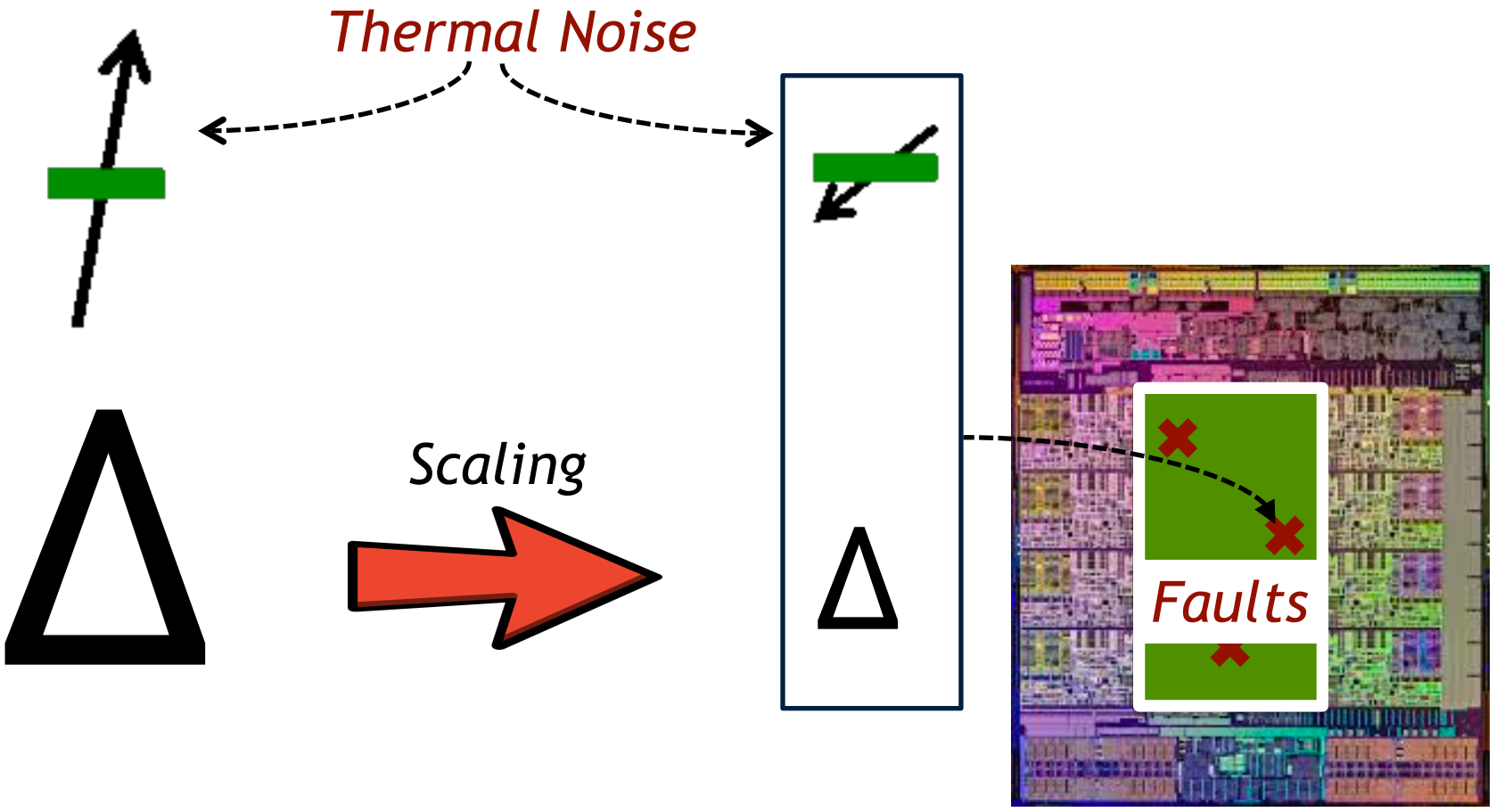
BACKGROUND



BACKGROUND

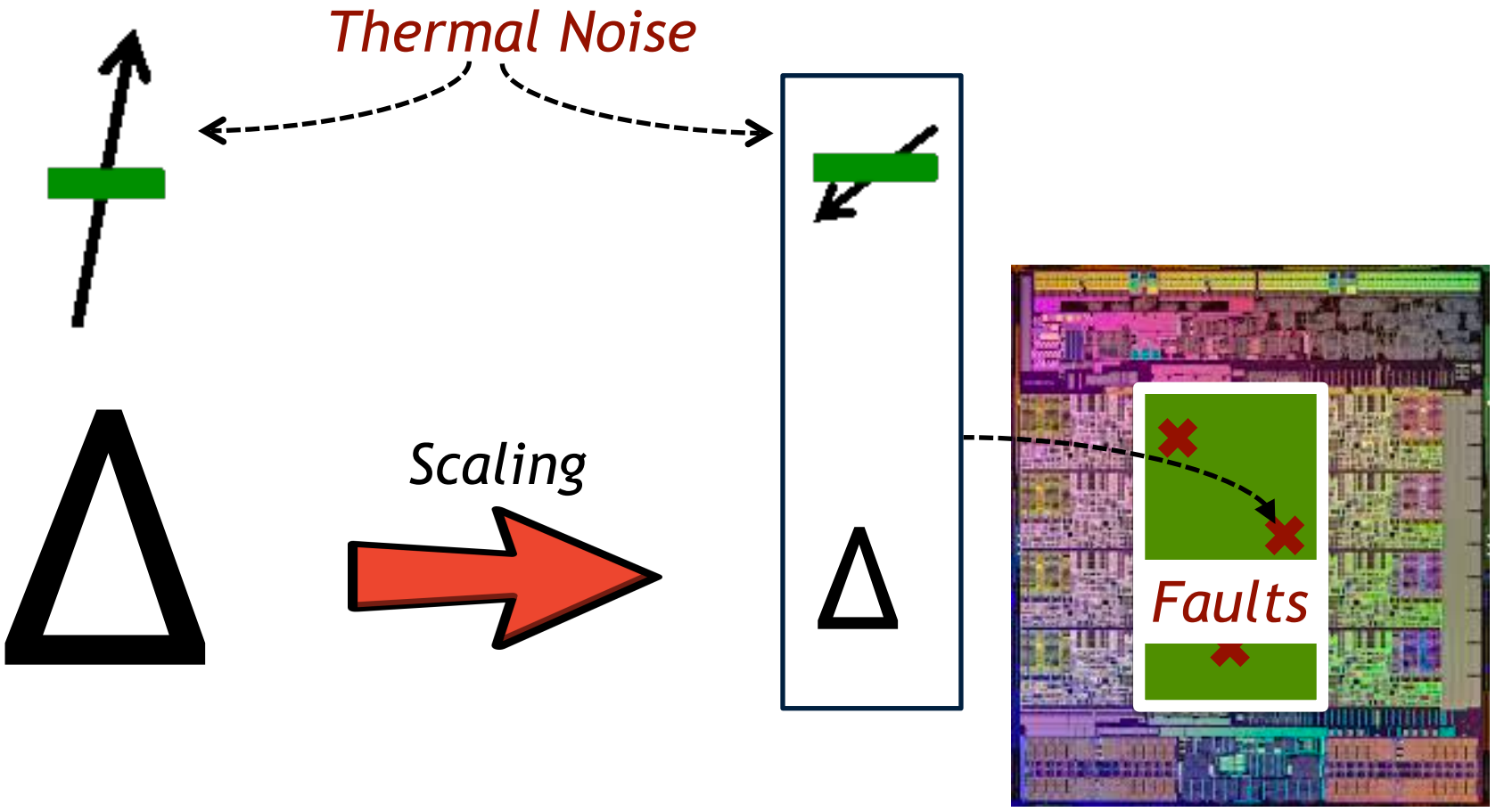


BACKGROUND



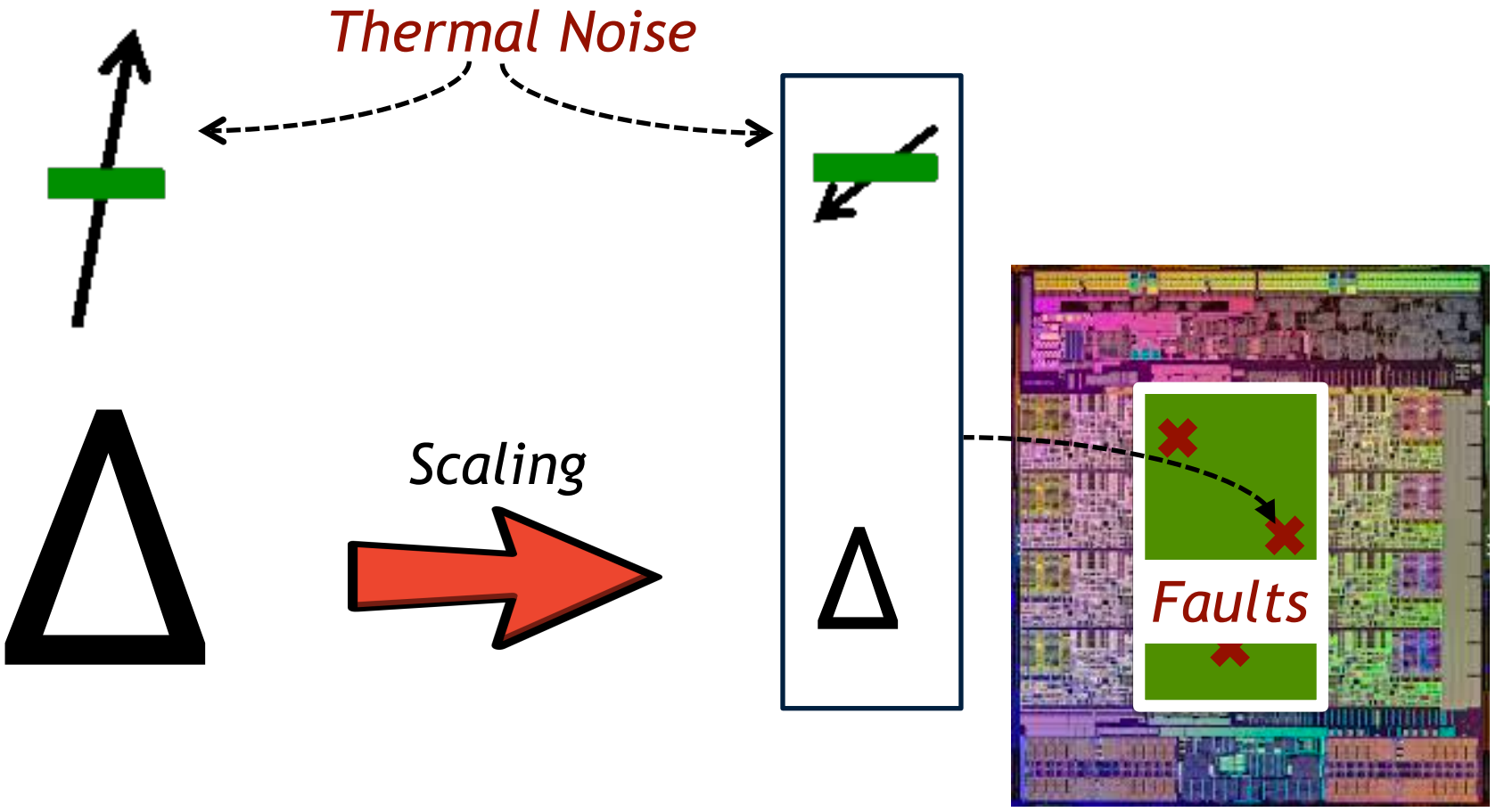
As STTRAM scales → High Rate of Retention Faults

BACKGROUND



As STTRAM scales → High Rate of Retention Faults

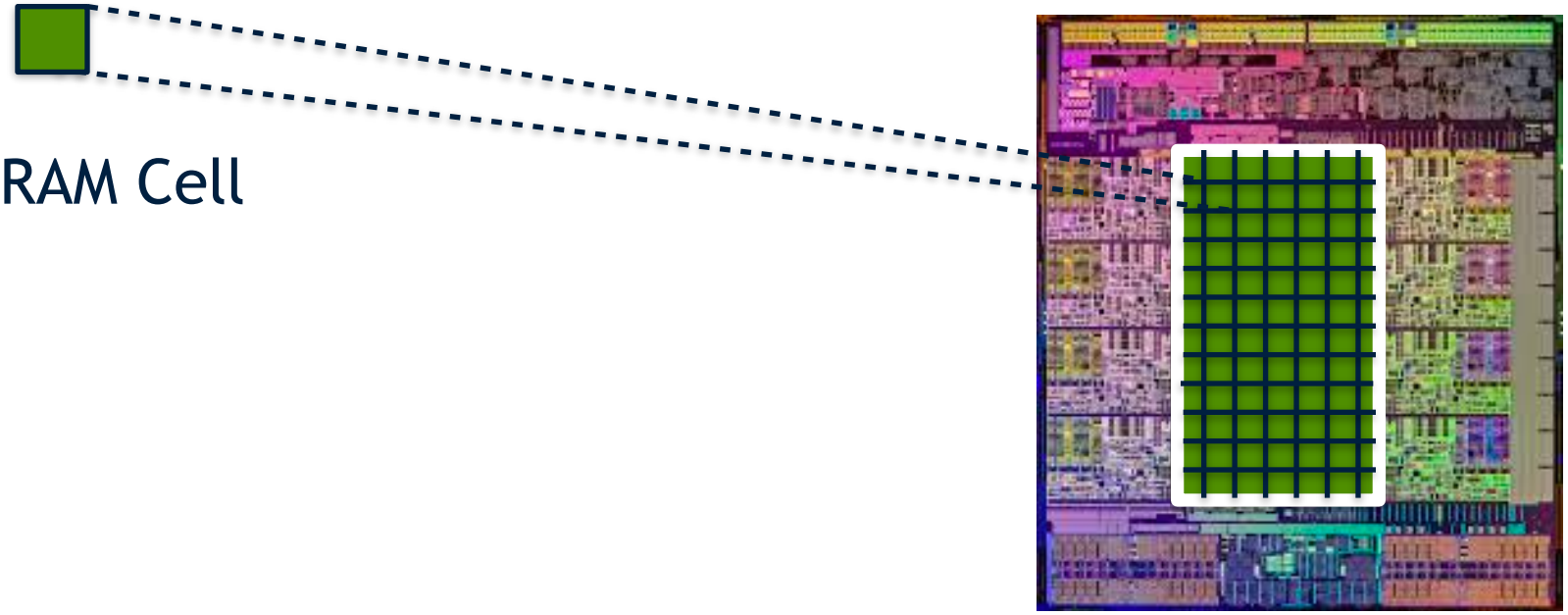
BACKGROUND



As STTRAM scales → High Rate of Retention Faults

BACKGROUND

Investigating the Mean Time to Failure (MTTF)

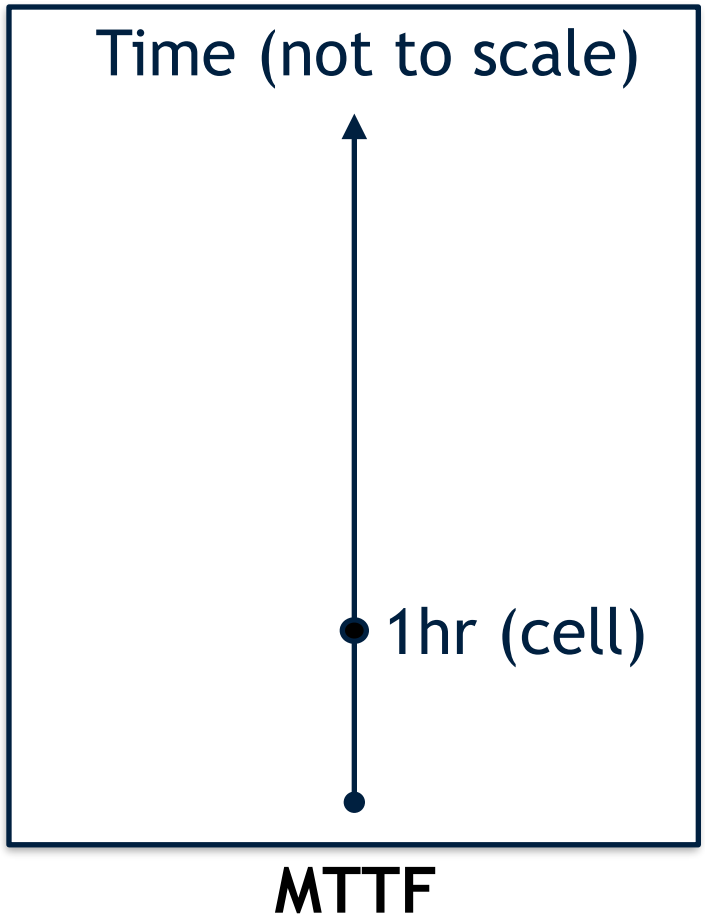


STTRAM Cell

BACKGROUND

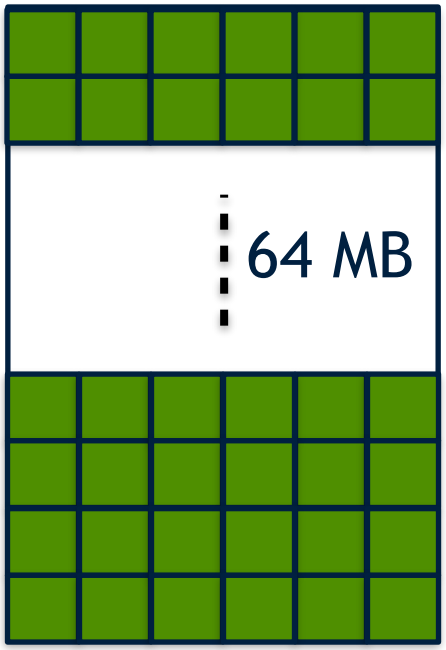
Investigating the Mean Time to Failure (MTTF)

STTRAM Cell
[22nm]

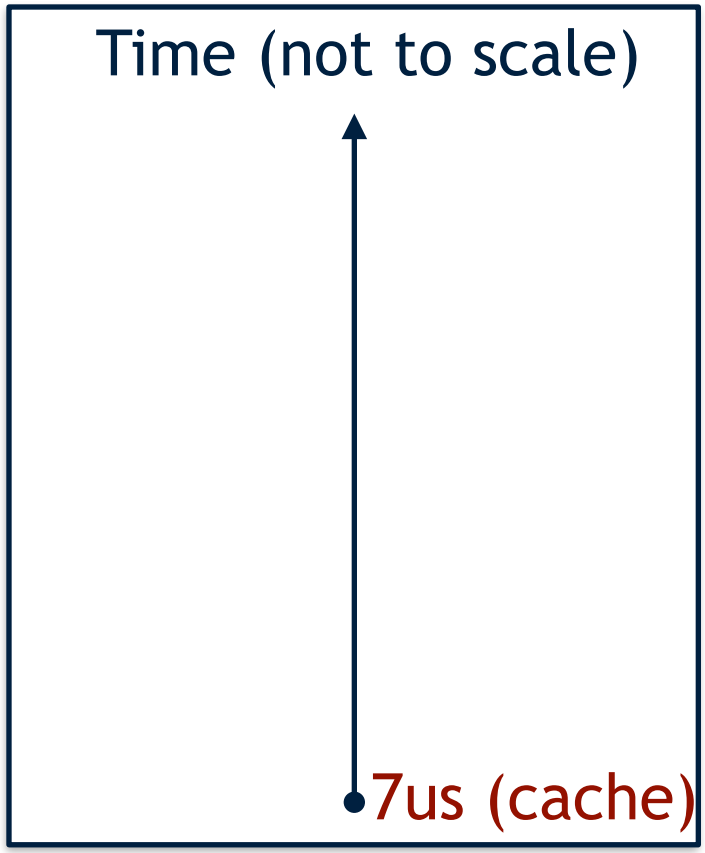


BACKGROUND

Investigating the Mean Time to Failure (MTTF)



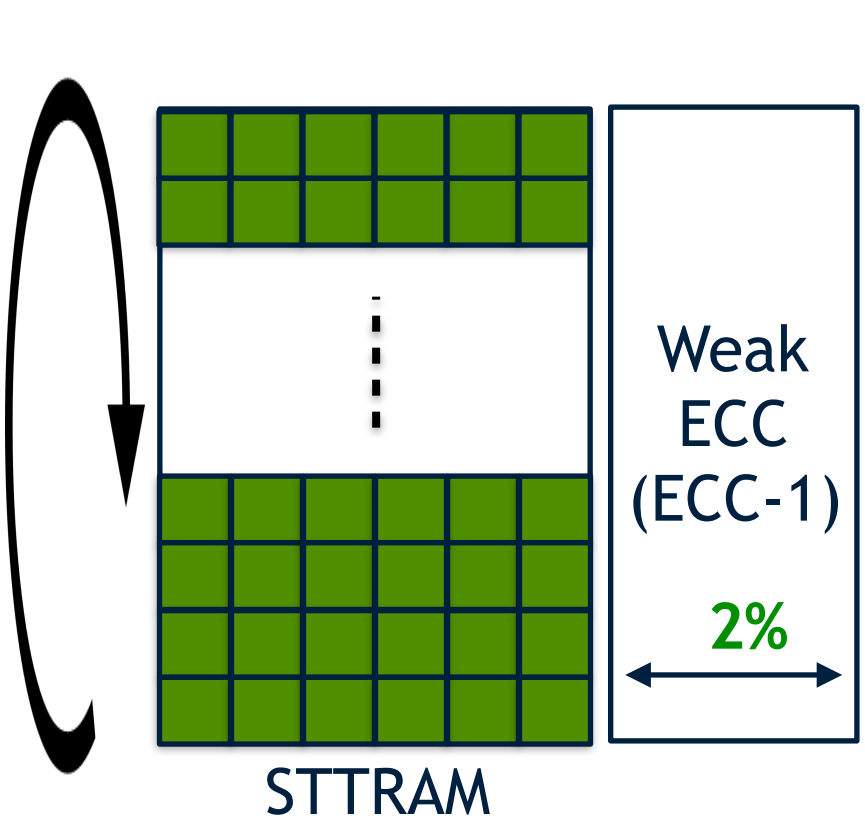
STTRAM Cache
[512 Million Cells]

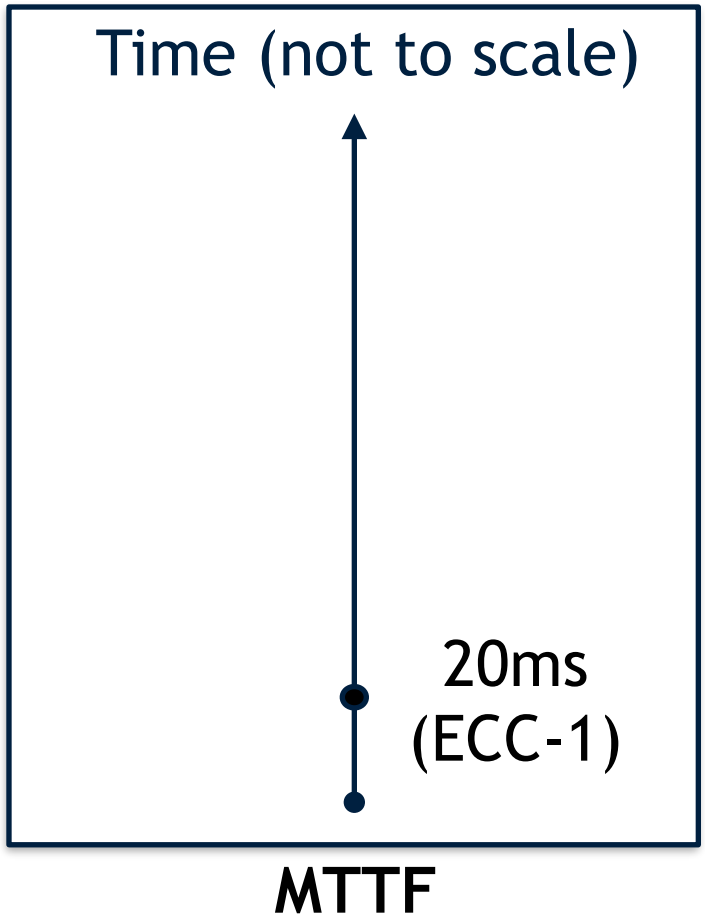


MTTF

BACKGROUND

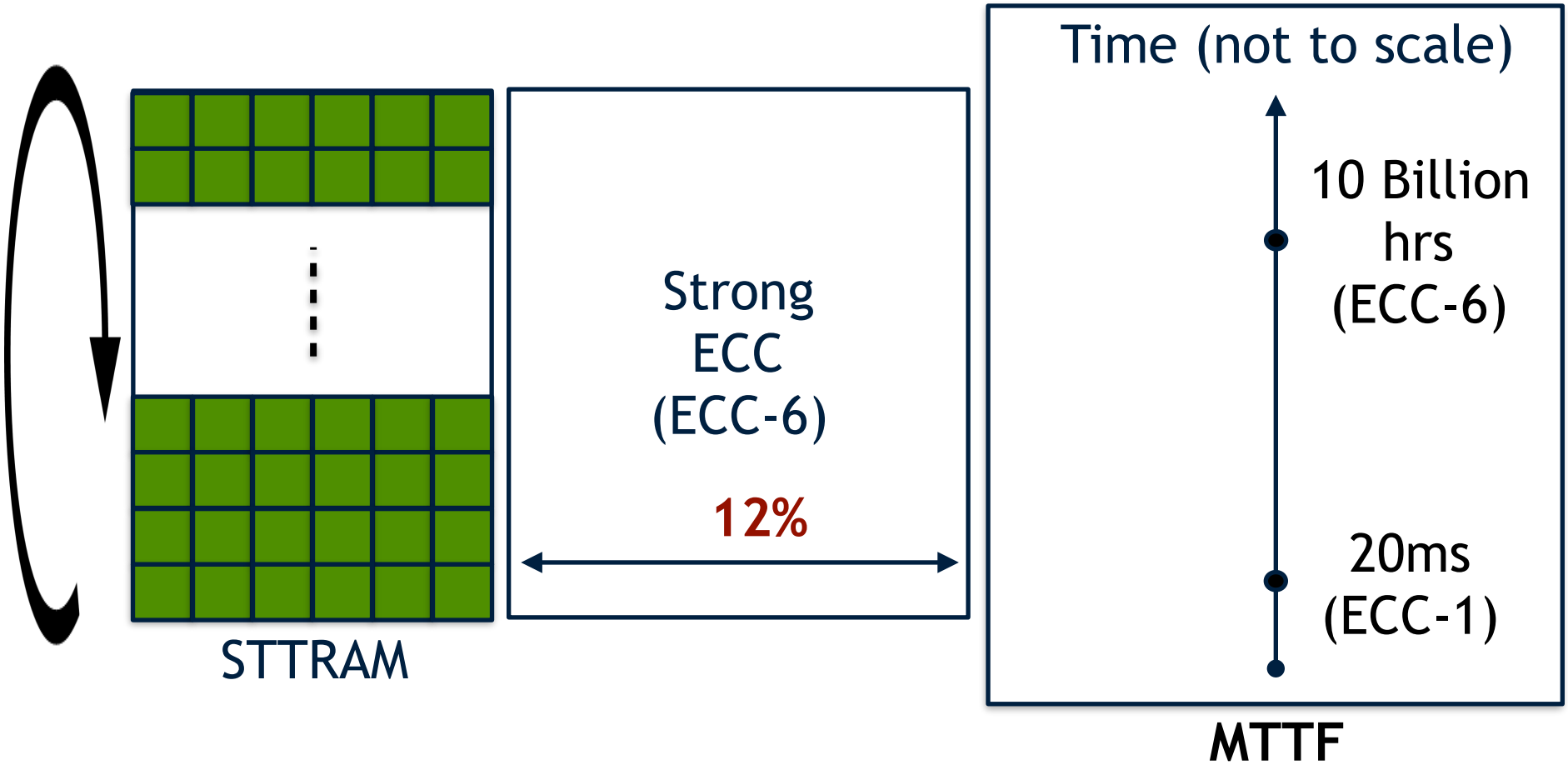
Scrubbing + Error Correcting Code (ECC) → High MTTF





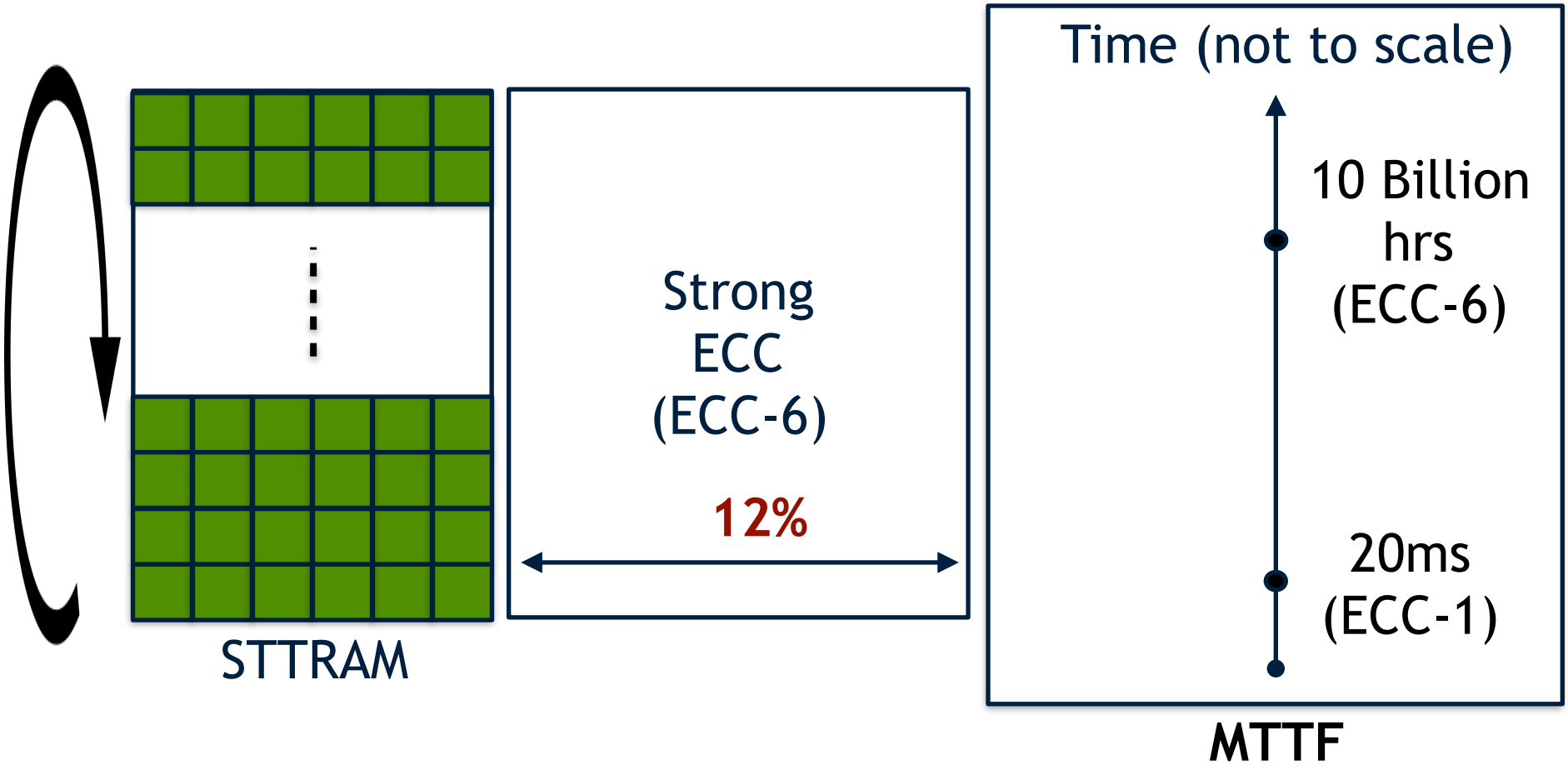
BACKGROUND

Scrubbing + Error Correcting Code (ECC) → High MTTF



BACKGROUND

Scrubbing + Error Correcting Code (ECC) → High MTTF



Ideally we need ECC-6 at the cost of ECC-1

OBSERVATIONS AND INSIGHT

Most faults (>99%) are 1-bit faults (common case)

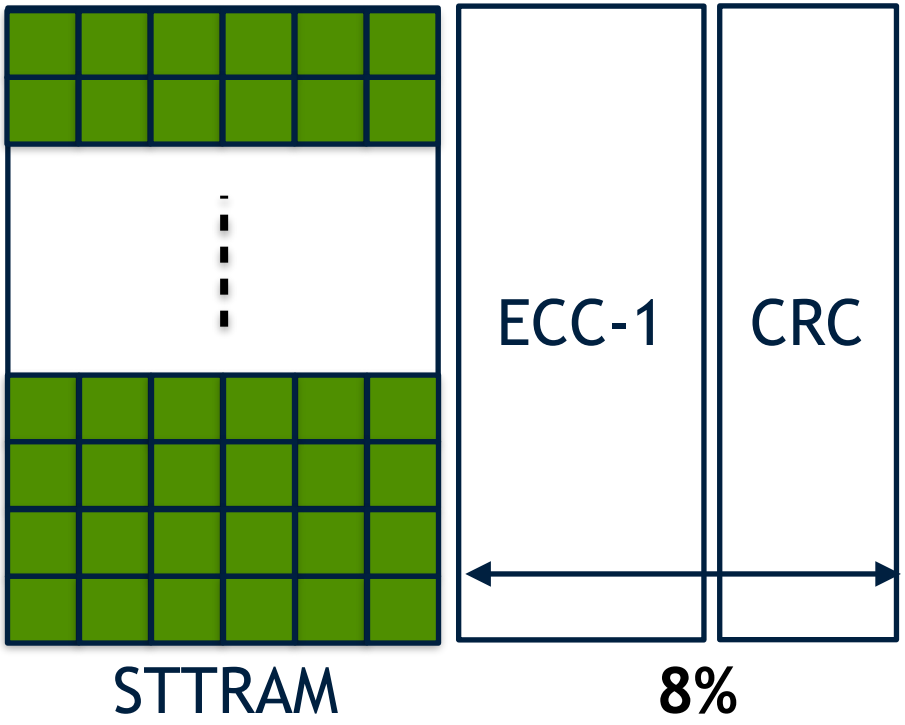
- Use ECC-1 for address the common case
- Low-cost for the common case

A mechanism to detect 2+ faults

- High-cost in uncommon case (<1% times)

SUDOKU

A low-cost mechanism that uses ECC-1 and CRC

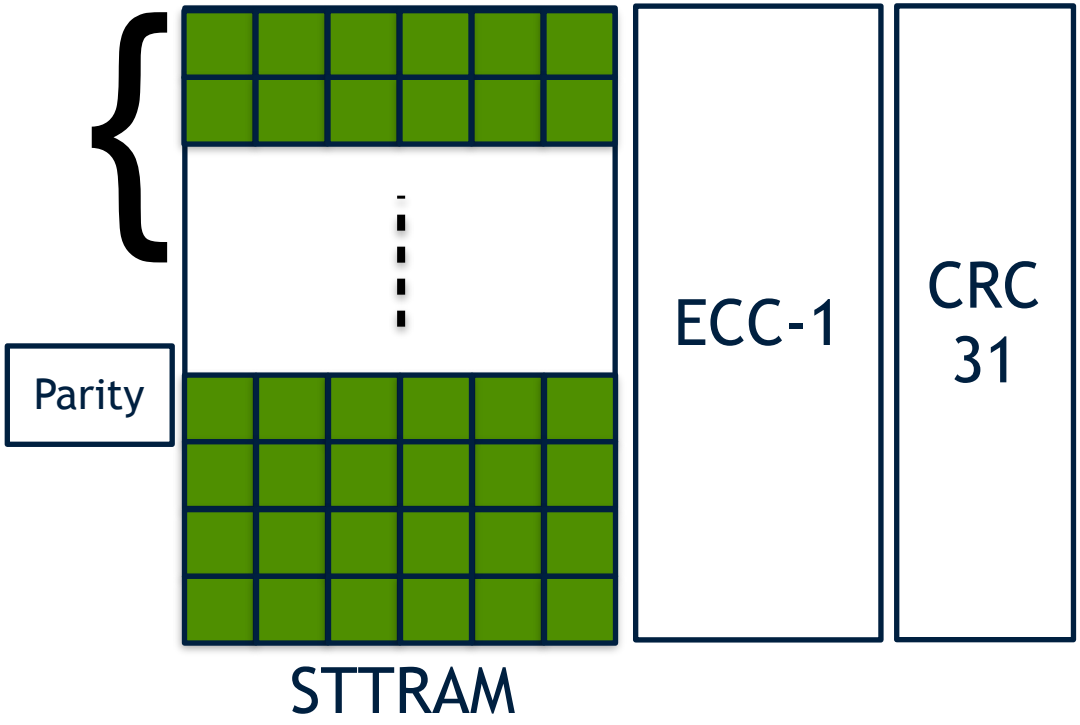


ECC-1 corrects 1-bit faults, CRC detects 2+ bit faults

SUDOKU-X

Split the STTRAM into regions: Use RAID-4

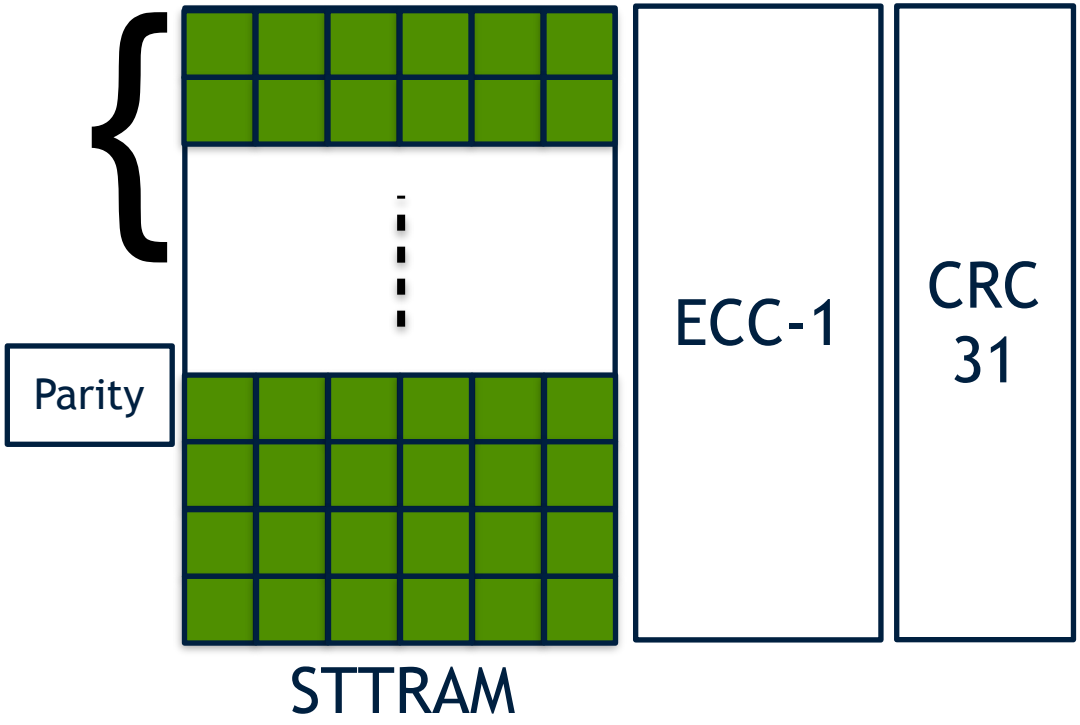
A SRAM parity structure to store parities per region



SUDOKU-X

Split the STTRAM into regions: Use RAID-4

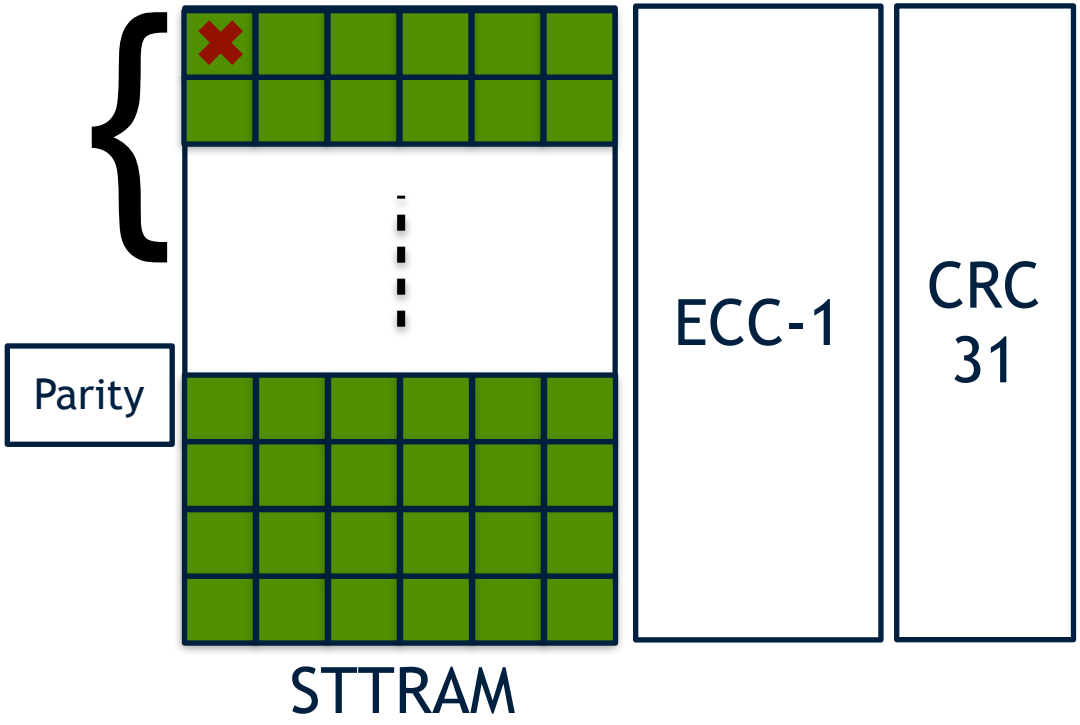
512 lines per region → Parity is 512x smaller



SUDOKU-X

Split the STTRAM into regions

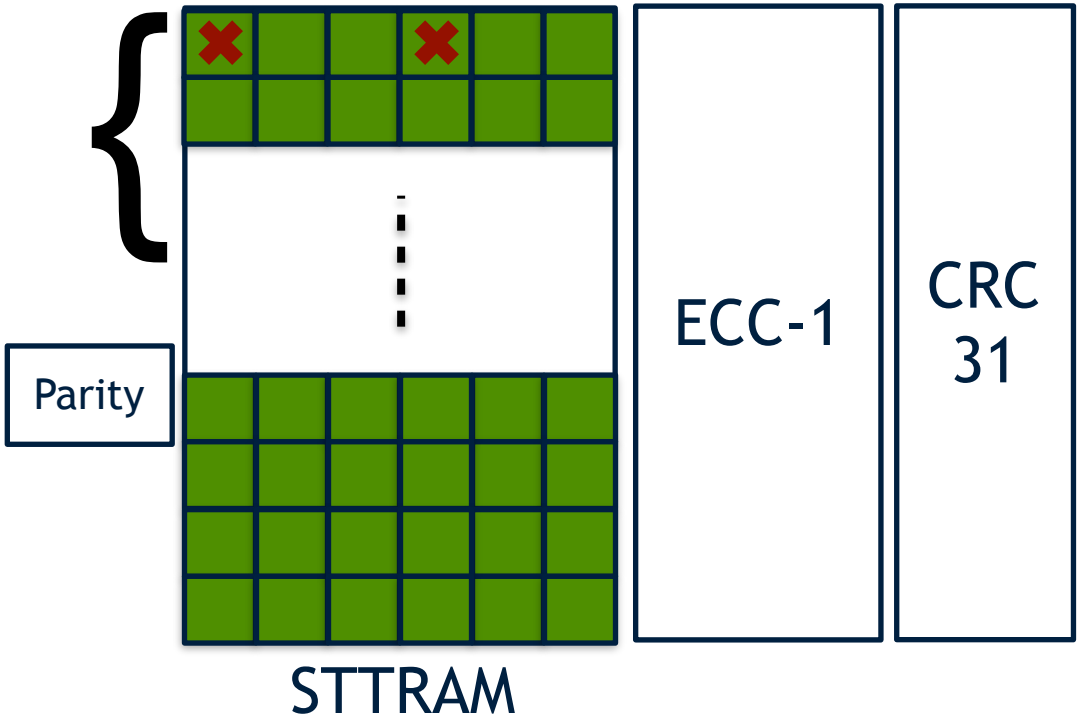
ECC-1 corrects single bit faults



SUDOKU-X

Split the STTRAM into regions

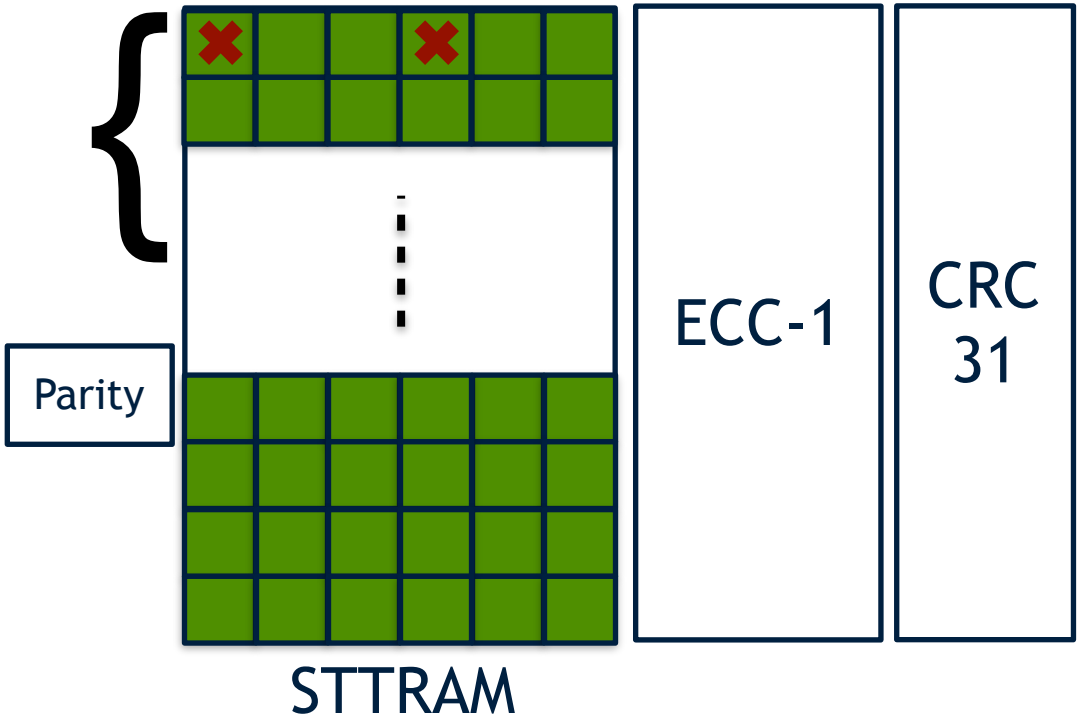
RAID-4 corrects multi-bit line faults: CRC-31 + Parity



SUDOKU-X

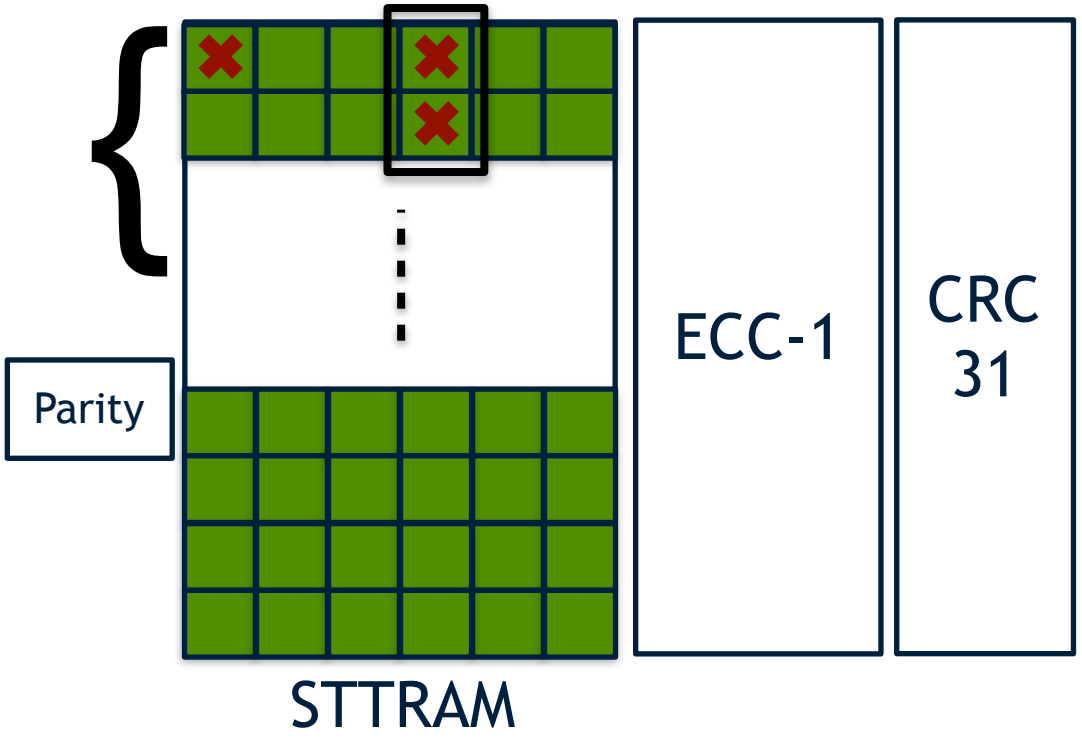
Split the STTRAM into regions

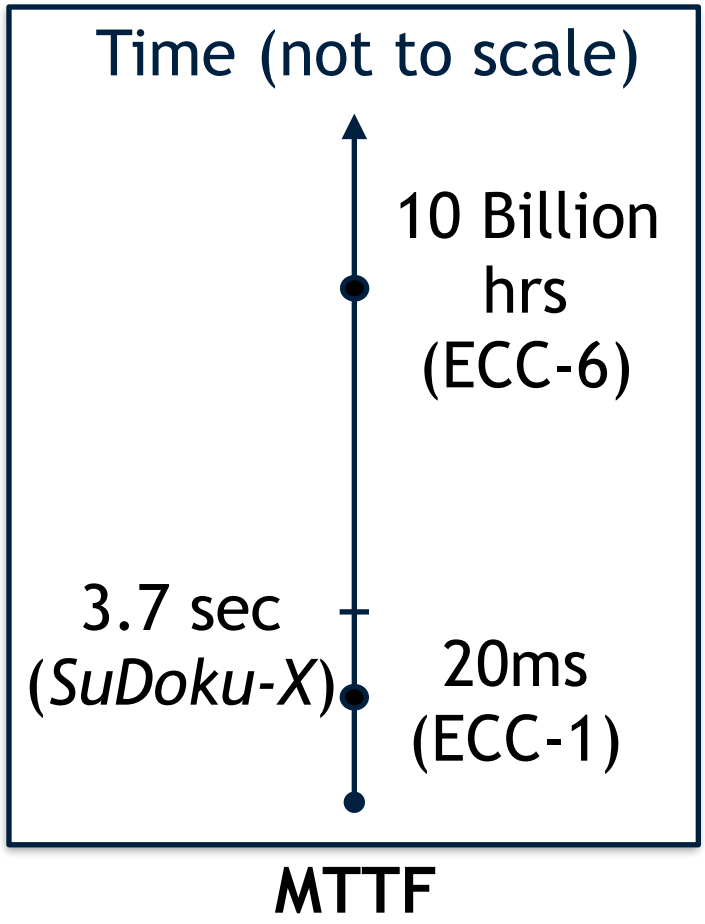
RAID-4 corrects multi-bit line faults: CRC-31 + Parity



SUDOKU-X

Fails when multiple faults correspond to the same parity bit



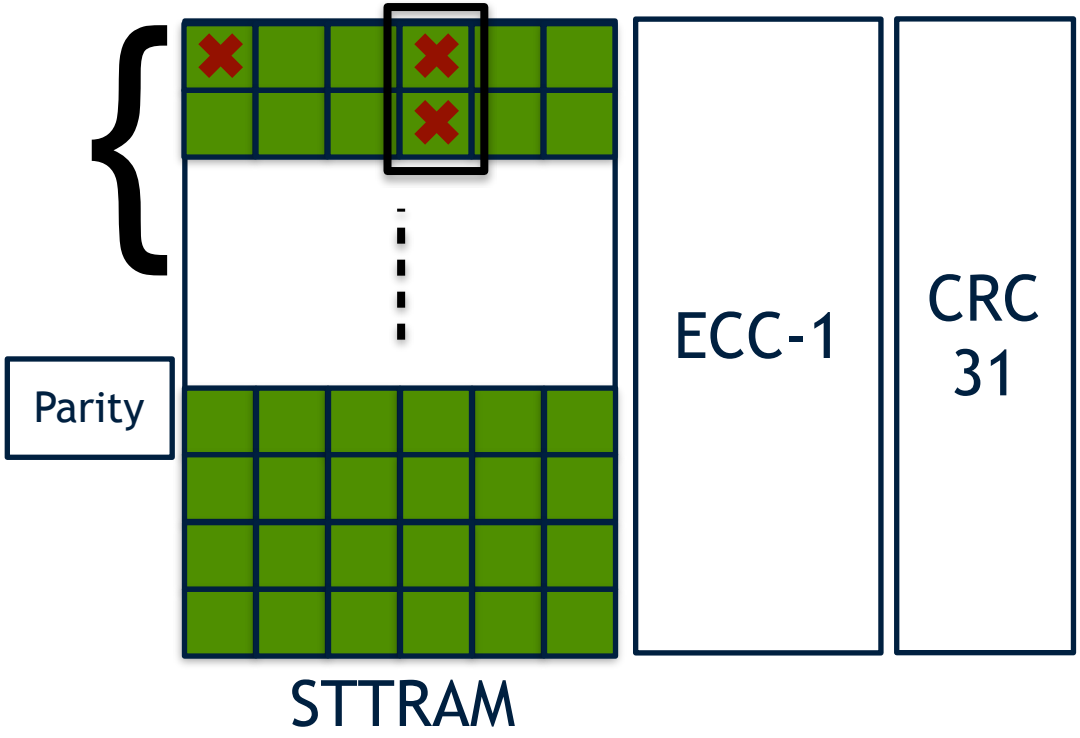


Improve MTTF by mitigating overlapping faults

SUDOKU-Y

In cases of overlapping faults: Flip and Retry

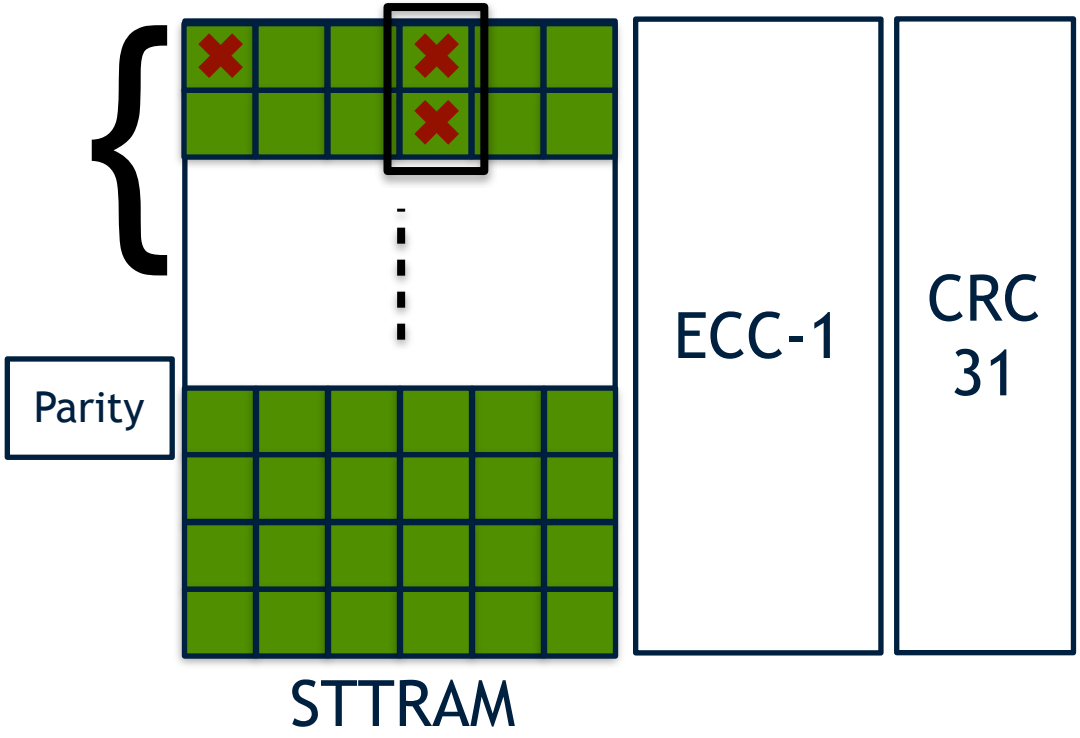
A mechanism to flip bits to fix multi-bit errors in a region



SUDOKU-Y

In cases of overlapping faults: Flip and Retry

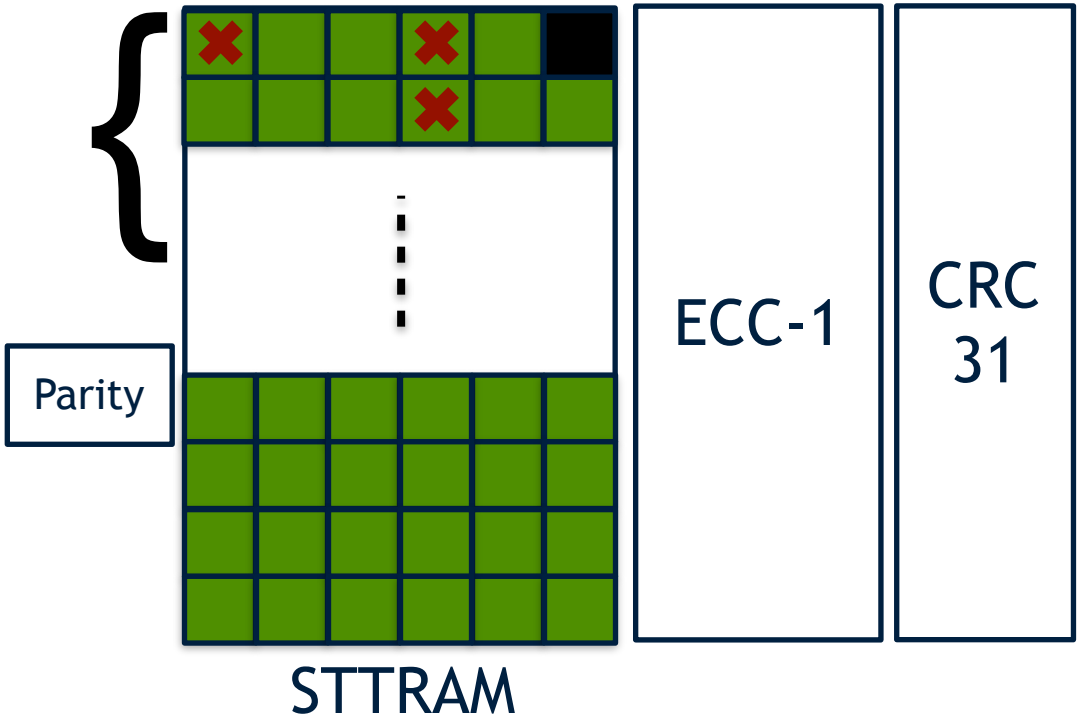
A mechanism to flip bits to fix multi-bit errors in a region



SUDOKU-Y

In cases of overlapping faults: Flip and Retry

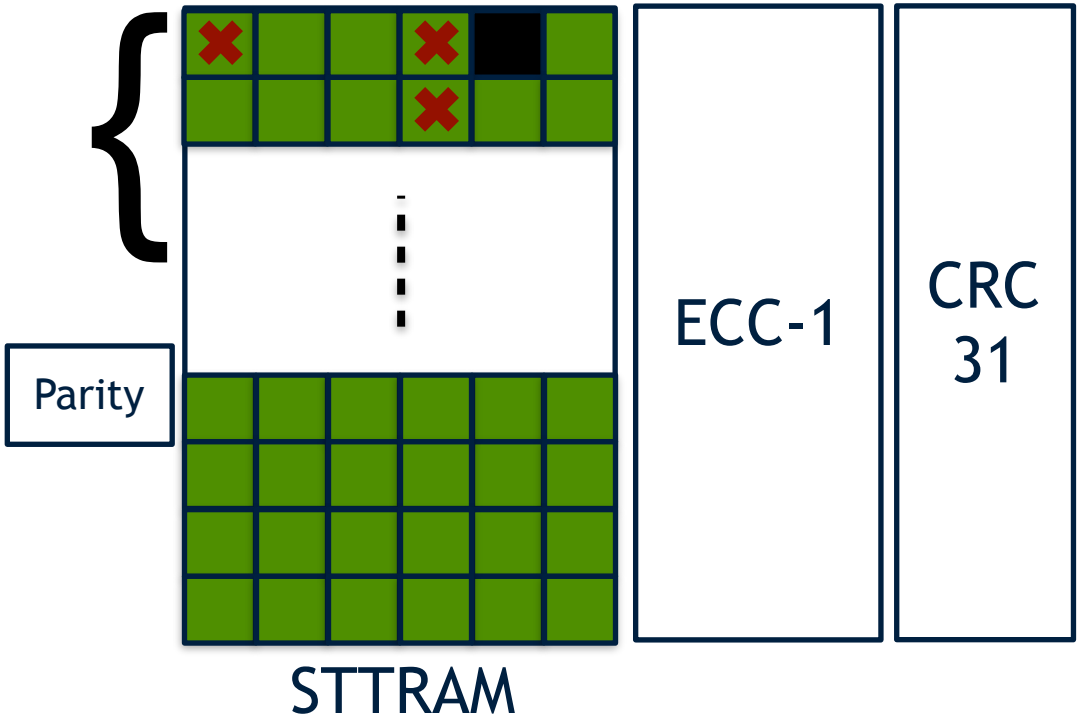
Sequentially flip each bit and check if CRC31 fails



SUDOKU-Y

In cases of overlapping faults: Flip and Retry

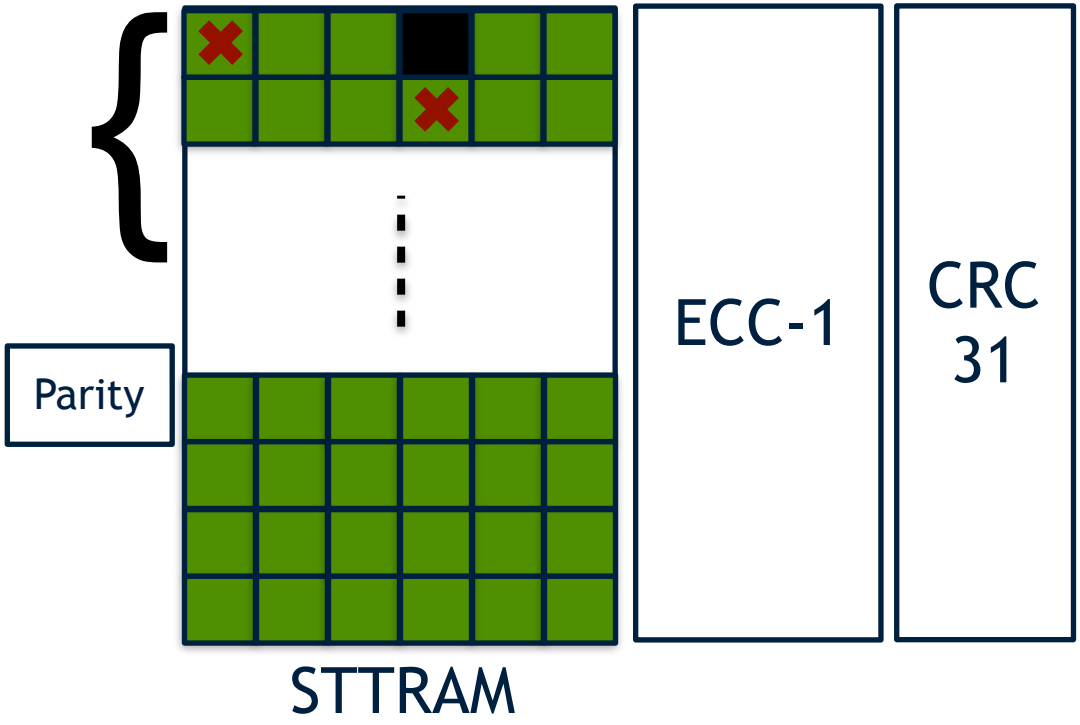
Sequentially flip each bit and check if CRC31 fails



SUDOKU-Y

In cases of overlapping faults: Flip and Retry

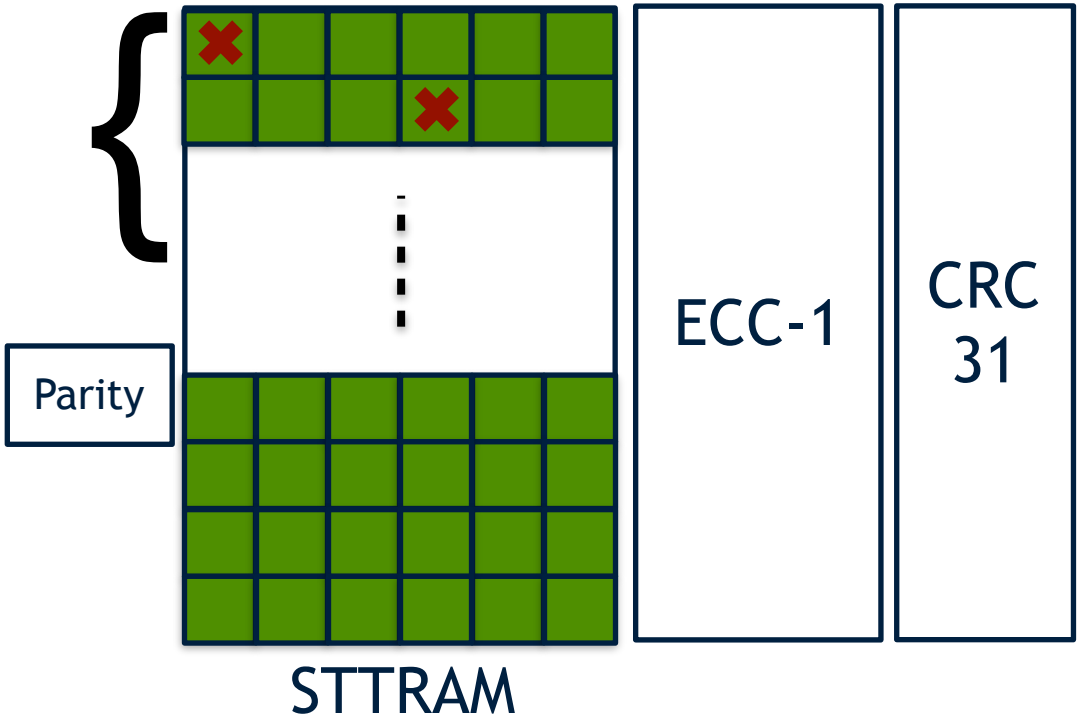
Sequentially flip each bit and check if CRC31 fails



SUDOKU-Y

In cases of overlapping faults: Flip and Retry

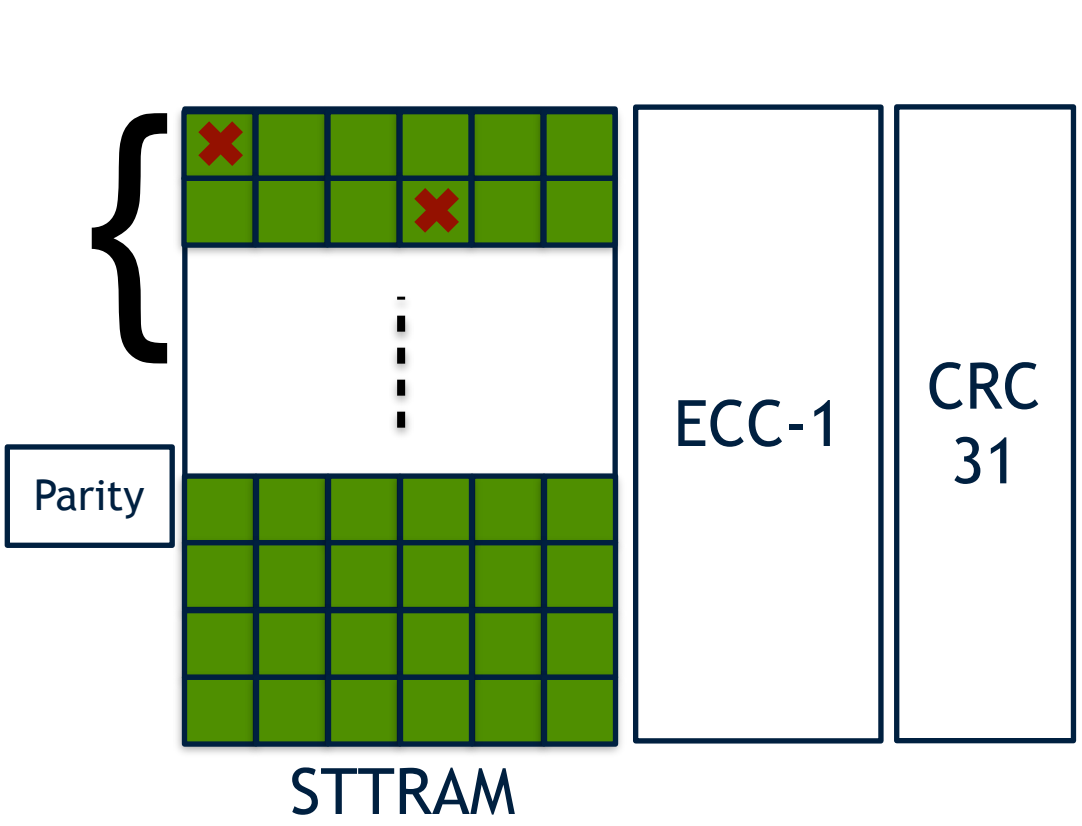
Sequentially flip each bit and check if CRC31 fails

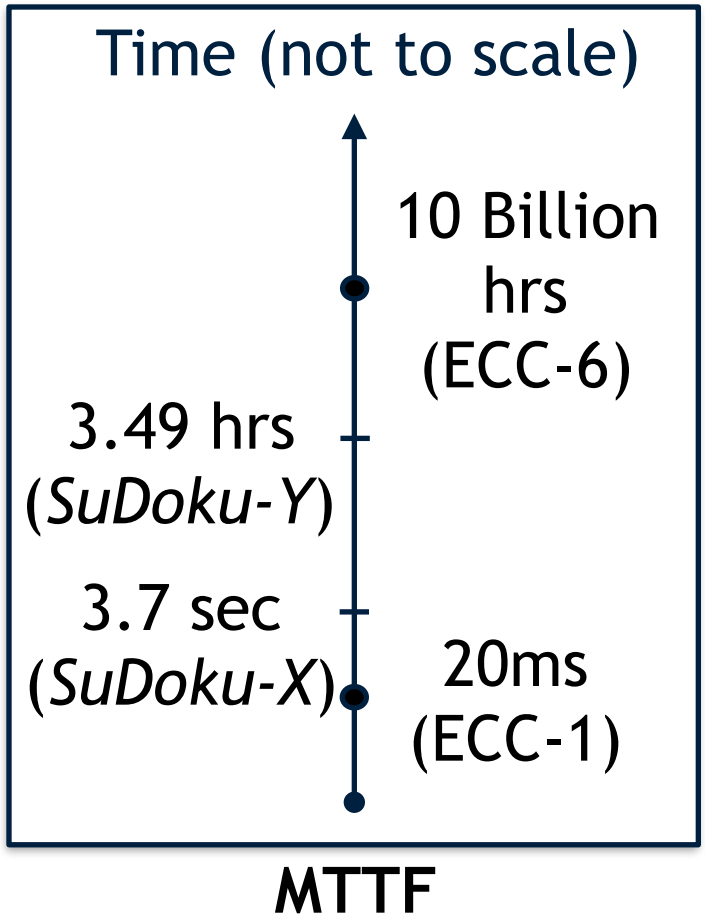


Use ECC-1 to fix the remaining 1-bit faults

SUDOKU-Y

In cases of overlapping faults: Flip and Retry



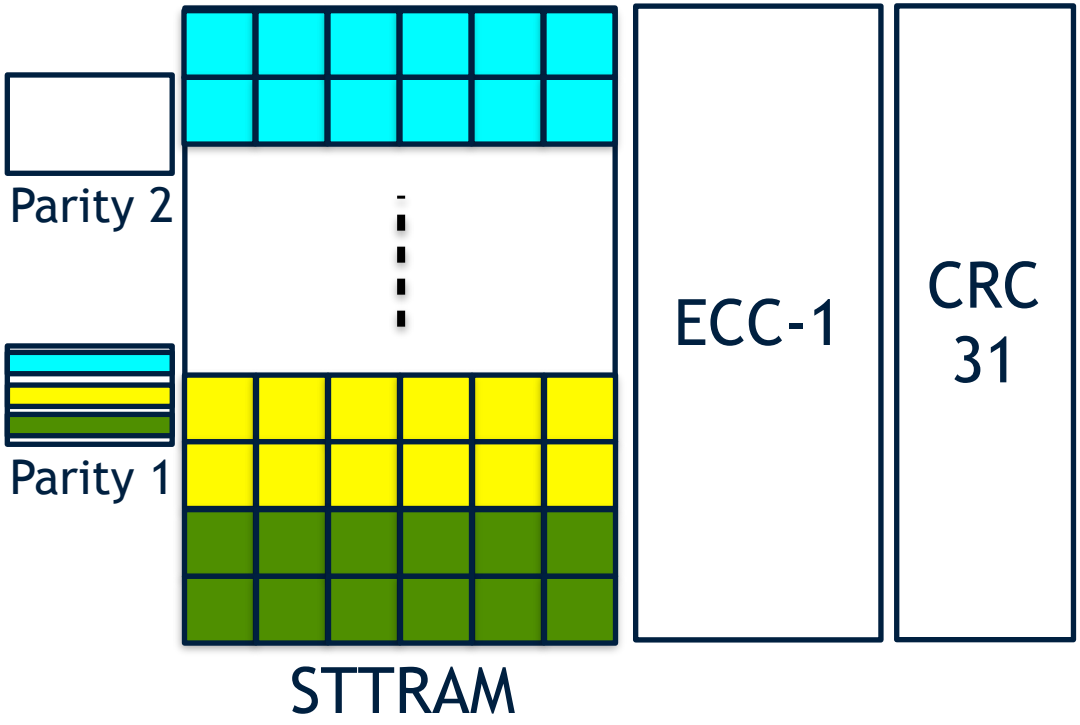


SuDoku-Y fails when multiple overlapping faults exist

SUDOKU-Z

Split the STTRAM into differently hashed regions

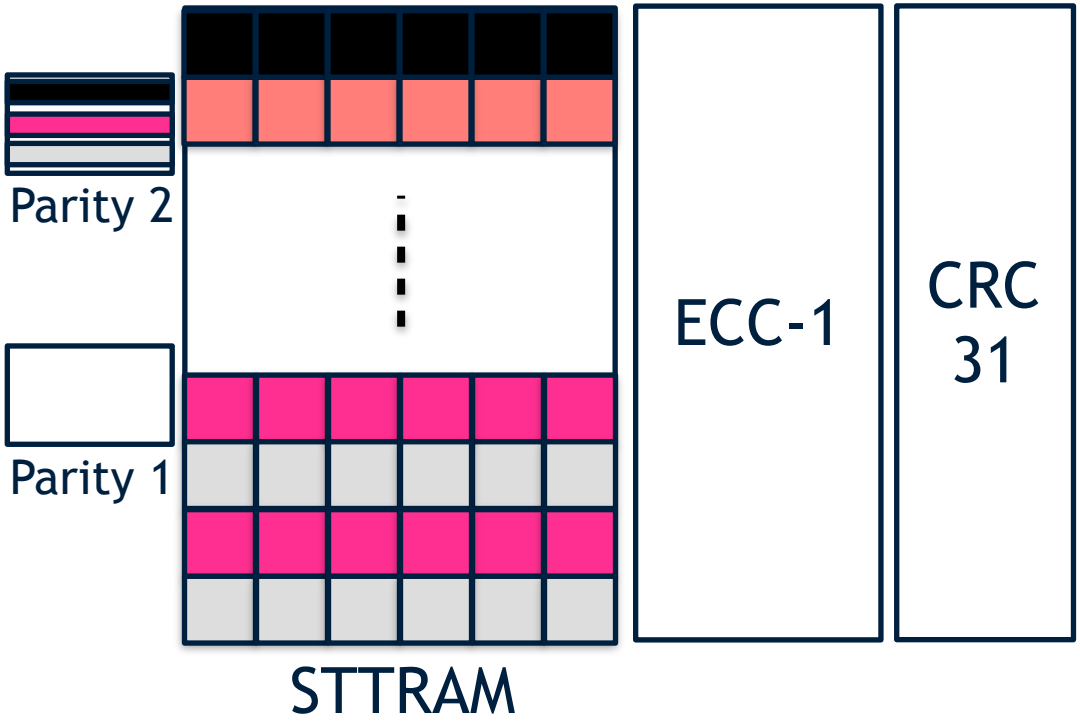
Create two parities based on these hashes



SUDOKU-Z

Split the STTRAM into differently hashed regions

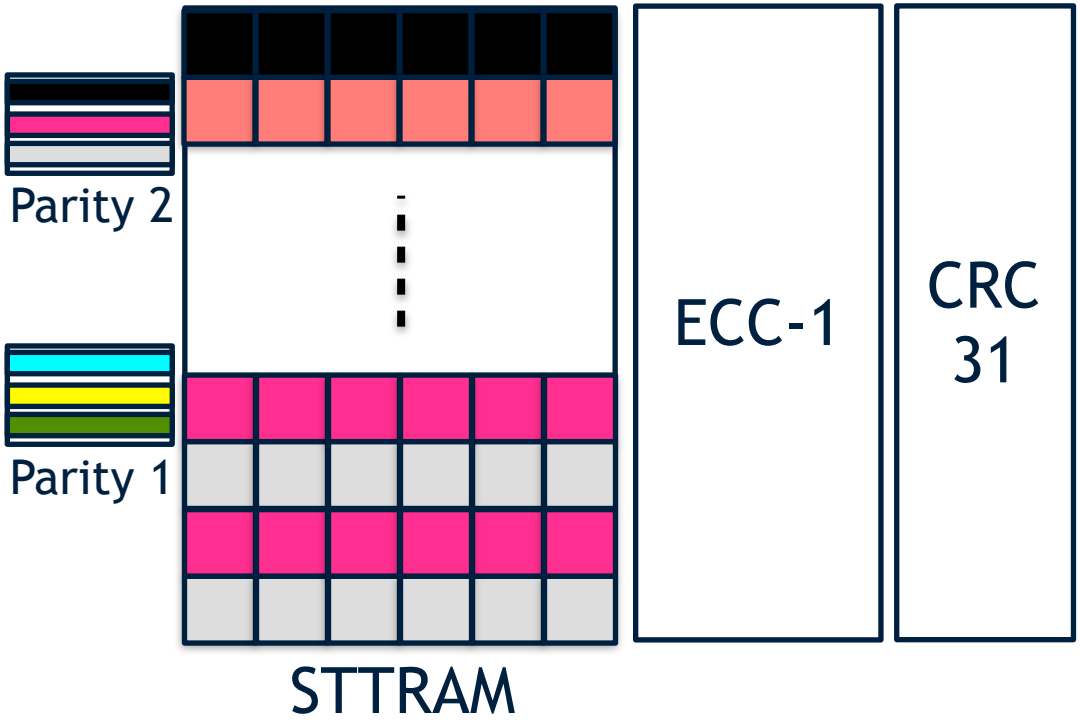
Create two parities based on these hashes



SUDOKU-Z

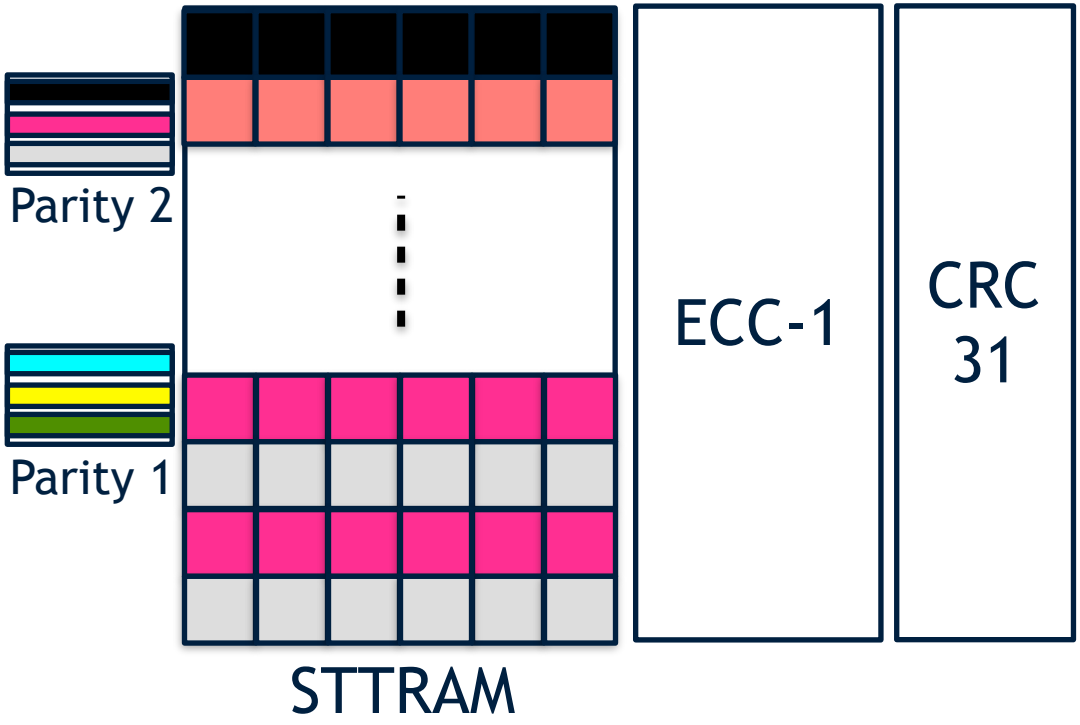
Split the STTRAM into differently hashed regions

If SuDoku-X + SuDoku-Y fails in Parity-1 → Try Parity 2



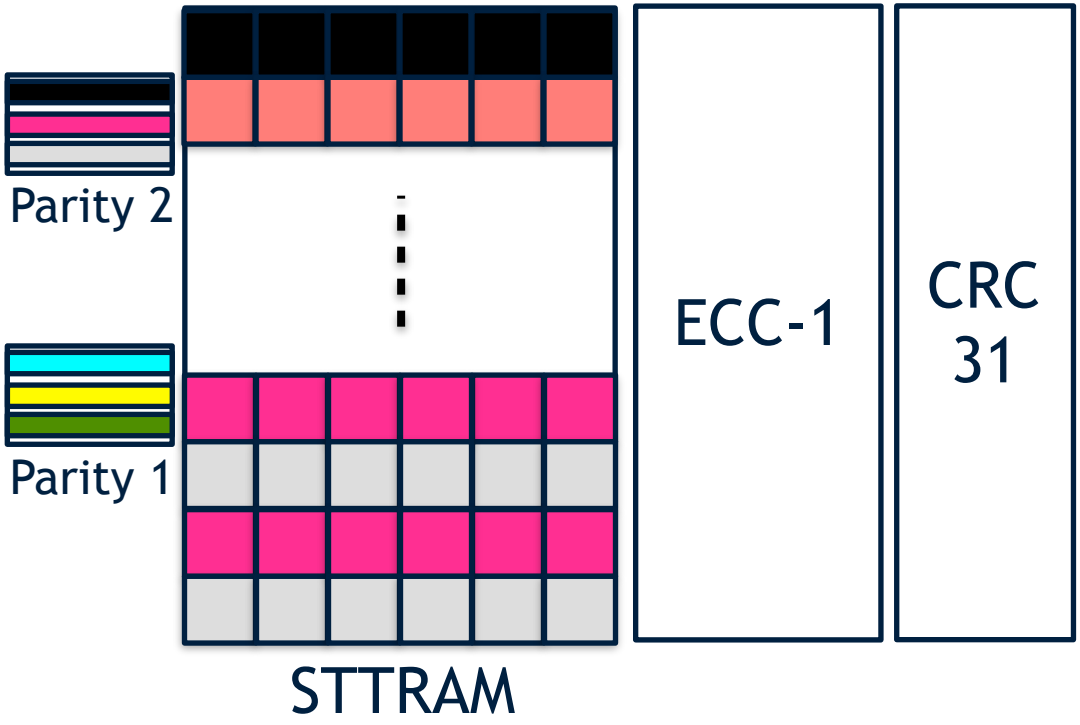
SUDOKU-Z

CRC-31 to find out if the SuDoku-X, SuDoku-Y succeeded



SUDOKU-Z

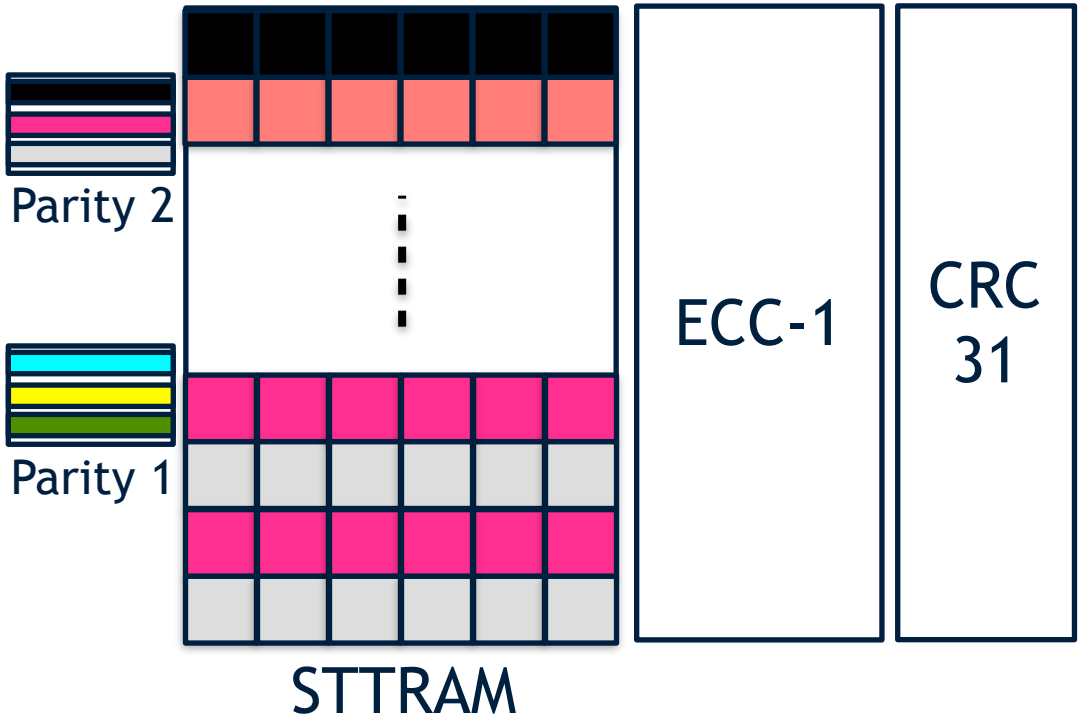
CRC-31 to find out if the SuDoku-X, SuDoku-Y succeeded

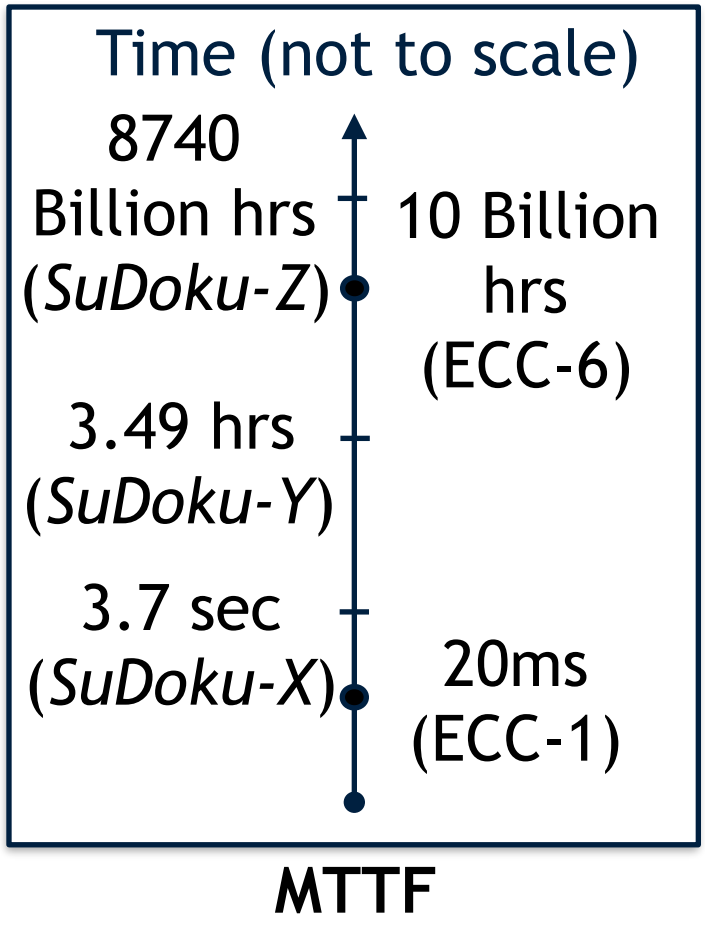


Strength is limited by CRC31 error detection rate

SUDOKU-Z

Split the STTRAM into differently hashed regions





SuDoku: 874x stronger than ECC-6 with low overheads

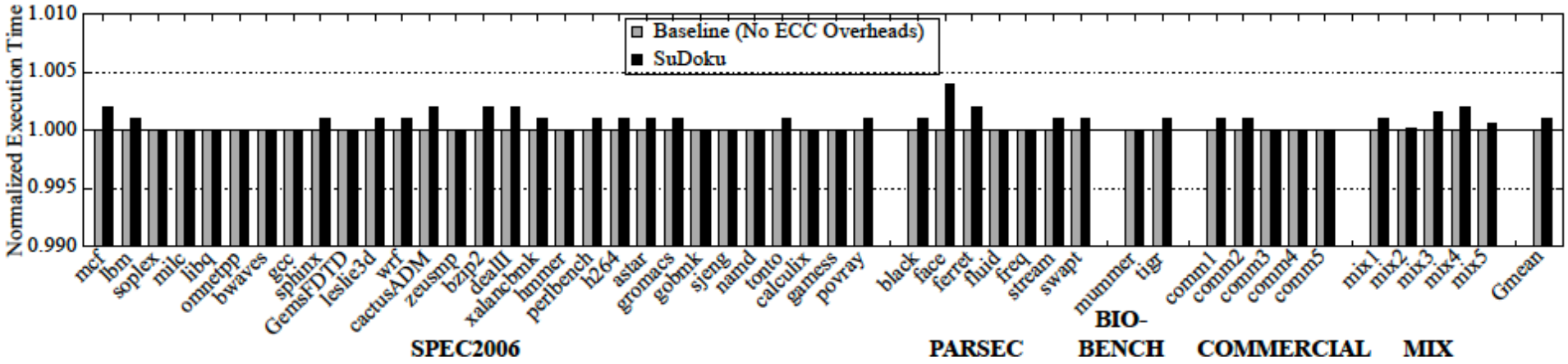
EVALUATION SETUP

USIMM Simulator: 8 OoO Cores, 64 MB STTRAM Cache

Retention Fault-Rate: 5.3×10^{-6}

Analytical models for reliability evaluations

PERFORMANCE RESULTS



Less than 1% performance overhead

INTUITION ON OVERHEADS

1. ECC-1 requires only 1 cycle to fix single-bit errors

2. RAID reads 512 lines → High Overhead
   - Fortunately, multi-bit errors occur infrequently: 4 times every 20ms
   - Total overhead is 16us every 20ms

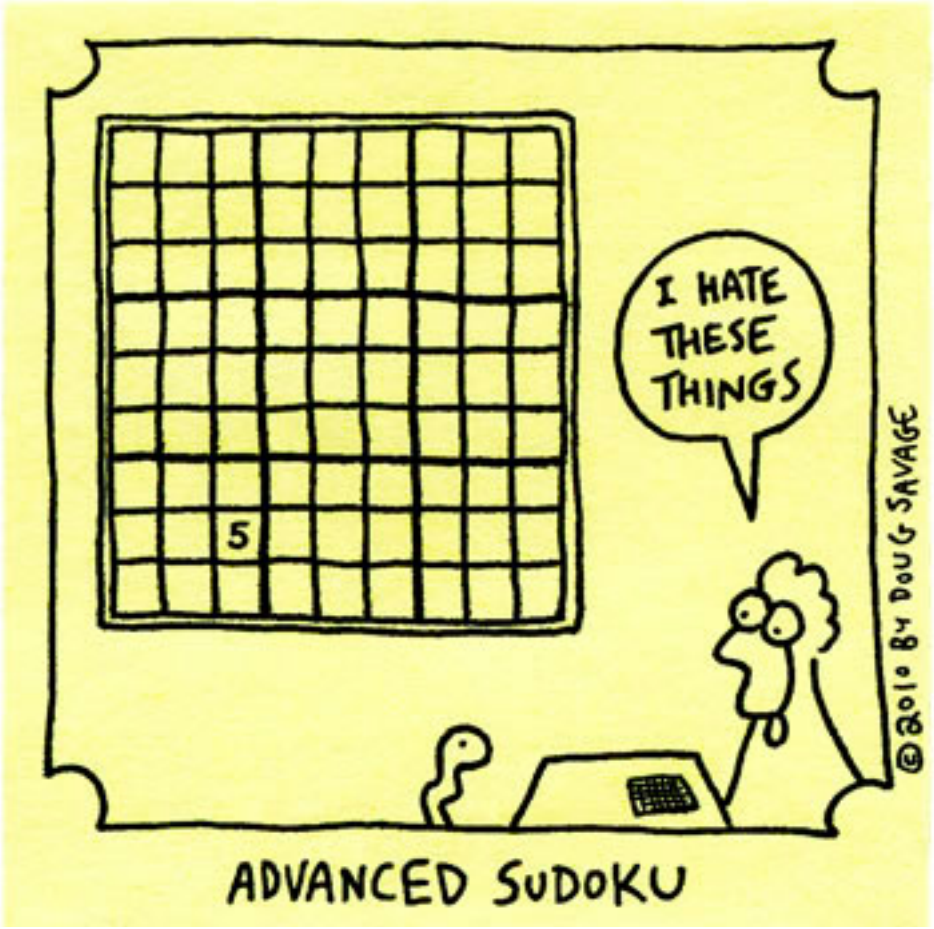
3. Parity based SRAMs are 512x smaller
4. Total area overhead from two parity arrays is 0.3%

Overall, SuDoku optimizes for the common case

SUMMARY

1. STTRAM can enable high density caches
2. STTRAM scales unreliably and may require costly ECC
3. To enable practical and efficient STTRAM we need strong ECC at lower costs.
4. SuDoku enables using ECC-1 (low cost) for the common case and mitigates overheads in enabling scalable STTRAM.
5. SuDoku uses strong ECC in the uncommon case and improves overall reliability

Thank You




THE UNIVERSITY OF BRITISH COLUMBIA