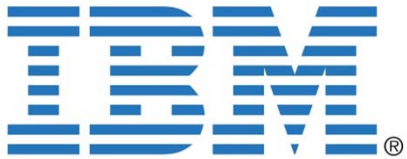
# Handling Resistance Drift in Phase Change Memory - Device, Circuit, Architecture, and System Solutions

Manu Awasthi[+], Manjunath Shevgoor[+], Kshitij Sudan[+], Rajeev Balasubramonian[+], Bipin Rajendran[‡], Viji Srinivasan[‡]

[+]University of Utah, [‡]IBM Research





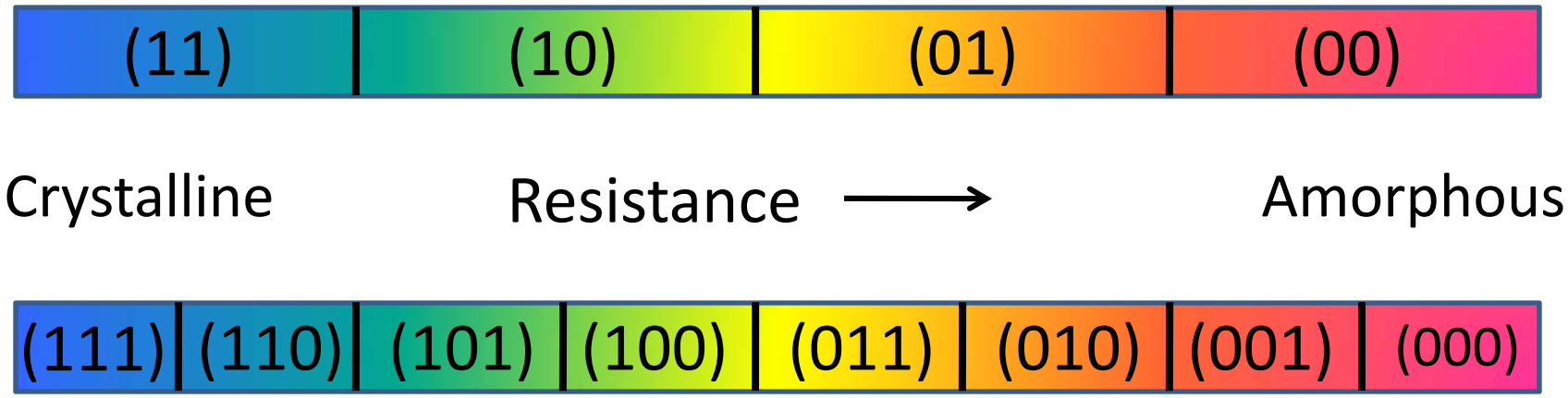
# Quick Summary

- Multi level cells in PCM appear imminent
- A number of proposals exist to handle hard errors and lifetime issues of PCM devices
- Resistance Drift is a lesser explored phenomenon
  - Will become increasingly significant as number of levels/cell increases – primary cause of "soft errors"
  - Naïve techniques based on DRAM-like refresh will be extremely costly for both latency and energy
  - Need to explore holistic solutions to counter drift

UTAH ARCH

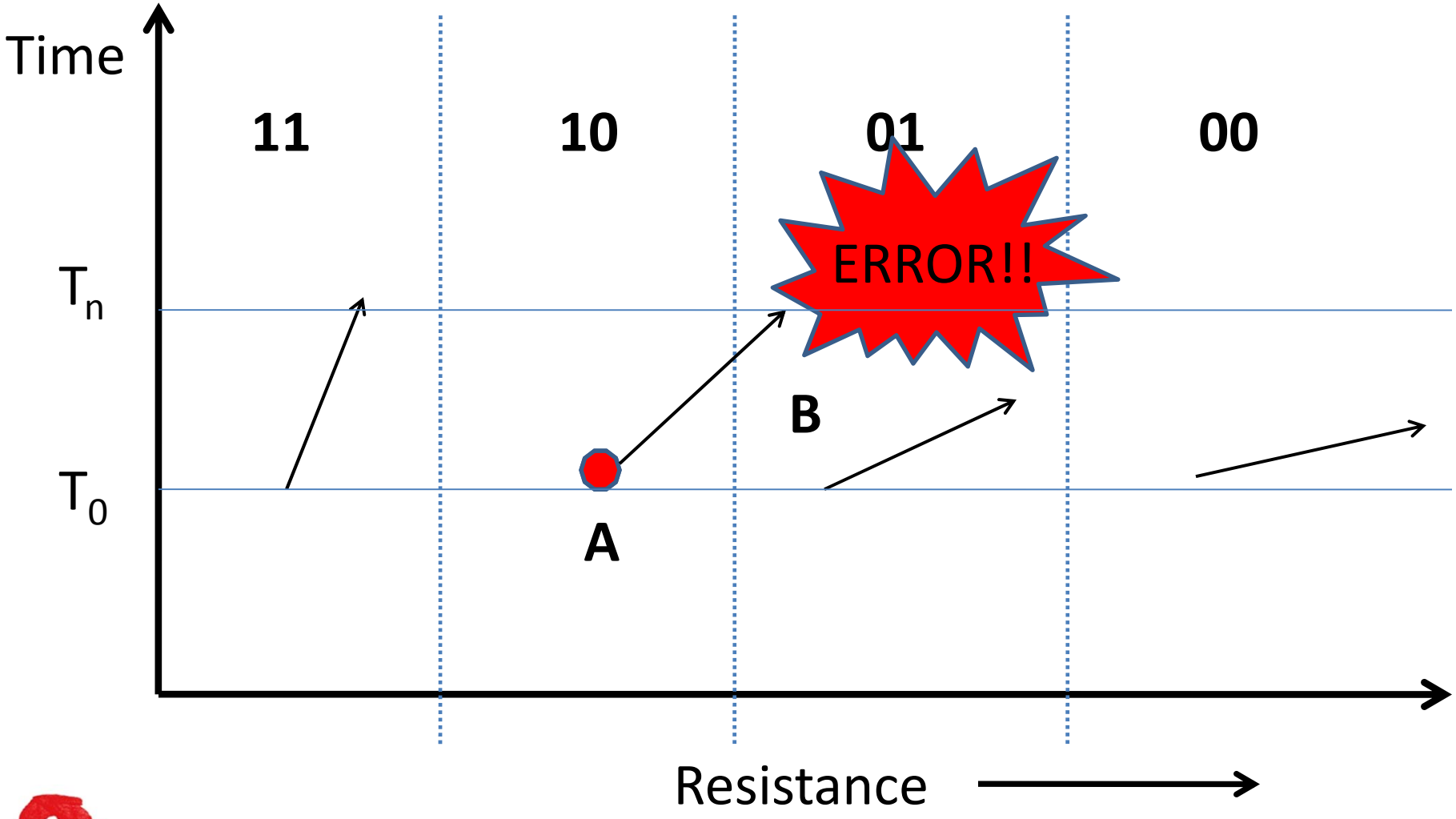
# Phase Change Memory - MLC

- Chalcogenide material can exist in crystalline or amorphous states
- The material can also be programmed into intermediate states
  - Leads to many intermediate states, paving way for **Multi Level Cells** (MLCs)




UTAH ARCH

# What is Resistance Drift?




UTAH ARCH

# Resistance Drift - Issues

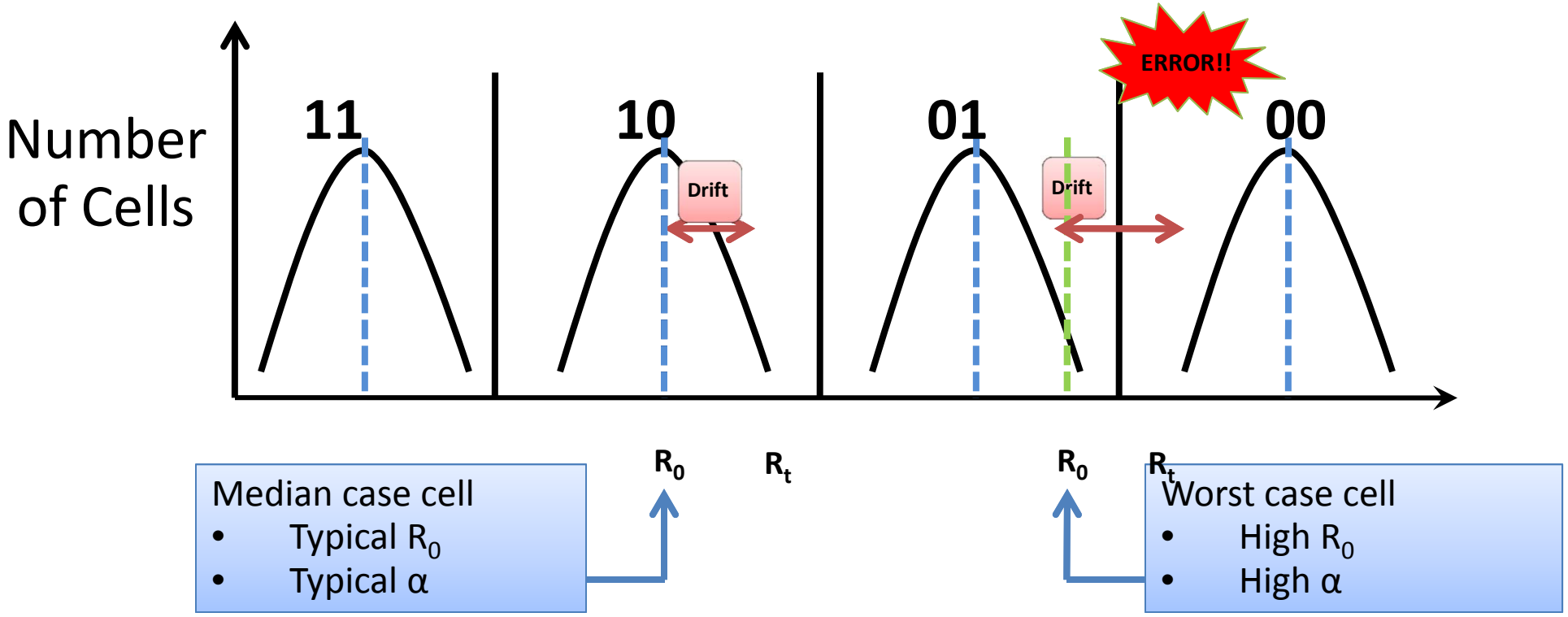
- Programmed resistance drifts according to power law equation -

$$R_{drift}(t) = R_0 \times (t)^{\alpha}$$

- $R_0$, $\alpha$ usually follow a Gaussian distribution
- Time to drift (error) depends on
  - Programmed resistance ($R_0$), and
  - Drift Coefficient ($\alpha$)
  - Is highly unpredictable!!

UTAH ARCH

# Resistance Drift - How it happens




Scrub rate will be dictated by the
**Worst Case $R_0$ and Worst Case α**

UTAH ARCH

# Resistance Drift Data

| Cell Type | Drift Time at Room temperature (secs) |
|---|---|
| Median 11 cell | $10^{499}$ |
| Worst 11 Case cell | $10^{15}$ |
| Median 10 cell | $10^{24}$ |
| **Worst Case 10 cell** | **5.94** |
| Median 01 cell | $10^{8}$ |
| **Worst Case 01 cell** | **1.81** |


UTAH ARCH

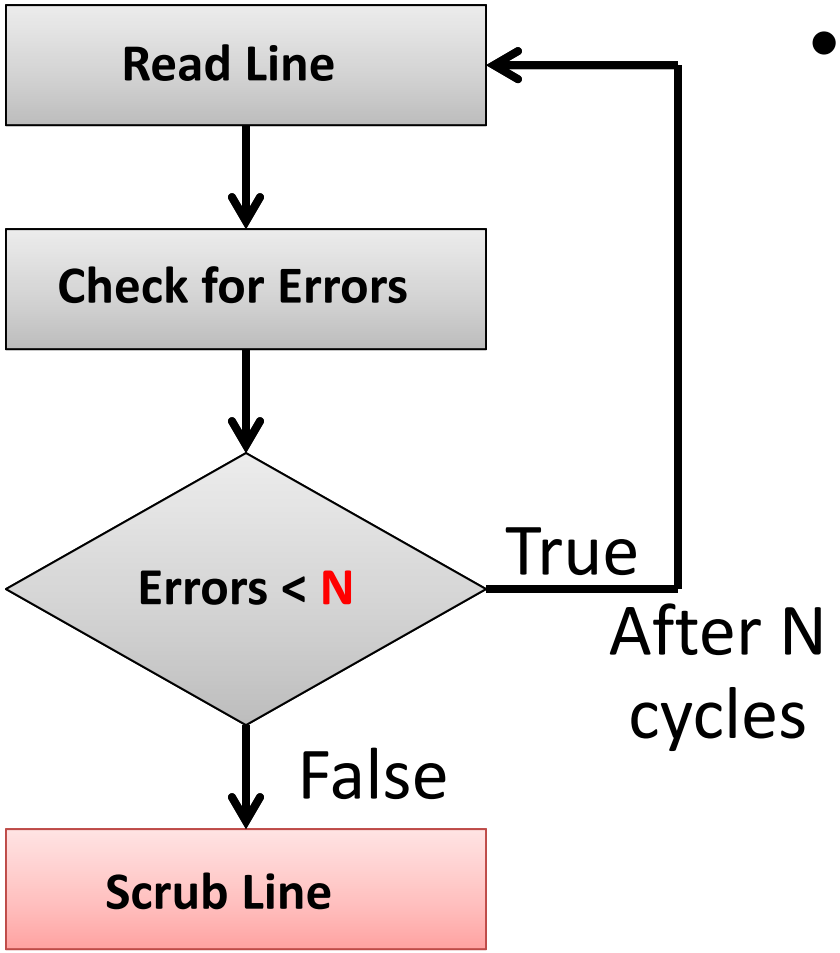
# Naïve Solution

- Drift resets with every cell reprogram (write)
- Leverage existing error correction mechanisms e.g. ECC - has its own drawbacks
- A Full Refresh (read-compare-write) is extremely costly in PCM
  - Each PCM write takes 100 - 1000ns
  - Writing to a 2-bit cell may consume as much as 1.6nJ
  - Requires 600 refreshes in parallel

**Refresh should be reactionary NOT precautionary!**

UTAH ARCH

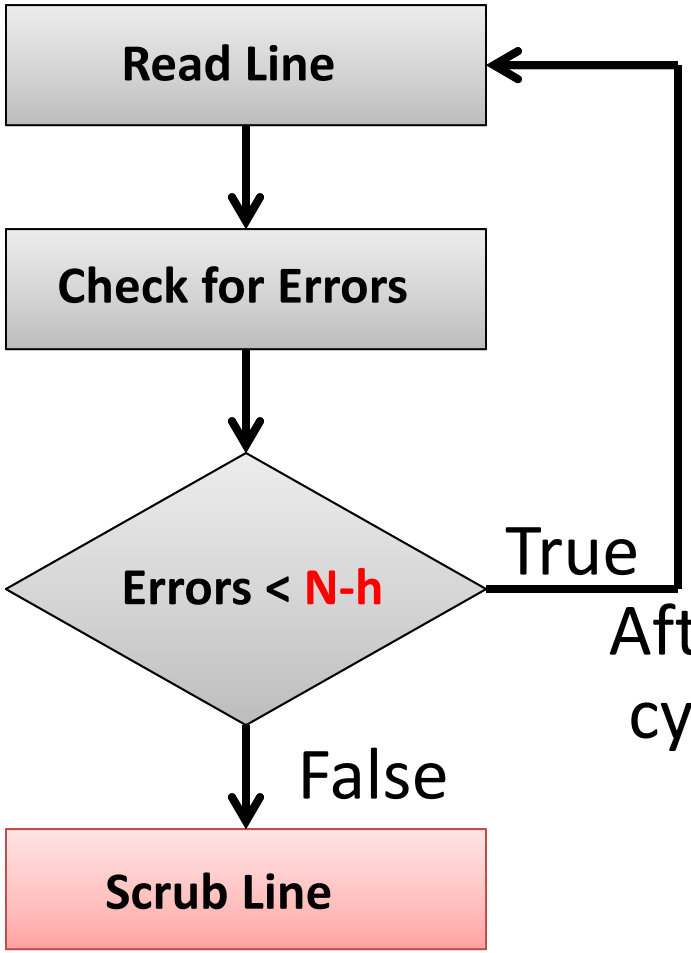
# Architectural Solutions - LARDD




- **L**ight **A**rray **R**eads for **D**rift **D**etection
  - Support for **N** Error-correcting, **N+1** error detecting codes assumed
  - Lines are read periodically and checked for correctness
  - Only after the number of errors reaches a threshold, scrubbing is performed

UTAH ARCH

# Architectural Solutions - Headroom




- **Headroom-h** scheme – scrub is triggered if **N-h** errors are detected
  - † Decreases probability of errors slipping through
  - – Increases frequency of full scrub and hence decreases life time
- Presents trade-off between **Hard** *a*nd **Soft** *e*rrors

UTAH ARCH

# Solutions Summary

**Architectural**

- Headroom schemes
  - Trade off between error rates and lifetime

**Device**

- Precise writes
- Guardbanding

**Circuit**

- Parity based technique
  - Makes common case faster
  - Reduces overheads

**System**

- Varying scrub rates
  - Accounts for changes in operating conditions
    - Temperature
    - Hard errors

UTAH ARCH

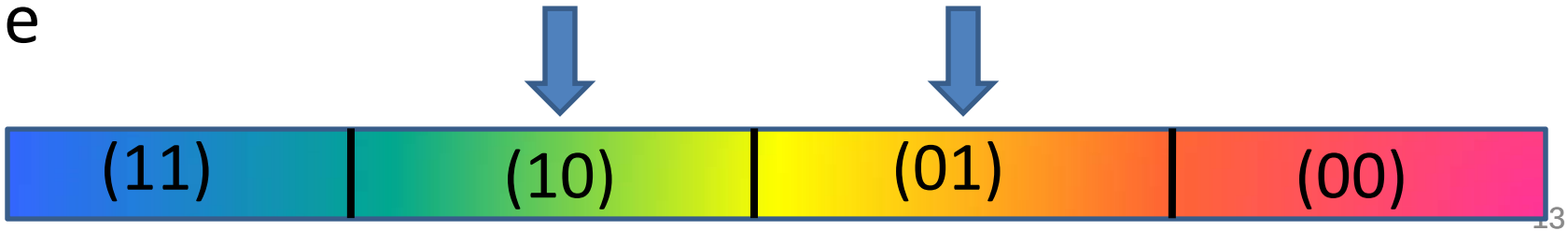
# System Level Solutions

- Dynamic events can affect reliability
  - Temperature increases can increase α and decrease drift time
  - Cell lifetime/wearout is also an issue
  - Soft error rate depends on prevalence of drift prone states
- These effects should be taken into account to dynamically adjust LARDD frequency
- Start with a low LARDD rate
  - Double rate when errors exceed pre-set threshold
  - Mark line as defunct when hard errors exceed pre-set threshold

UTAH ARCH

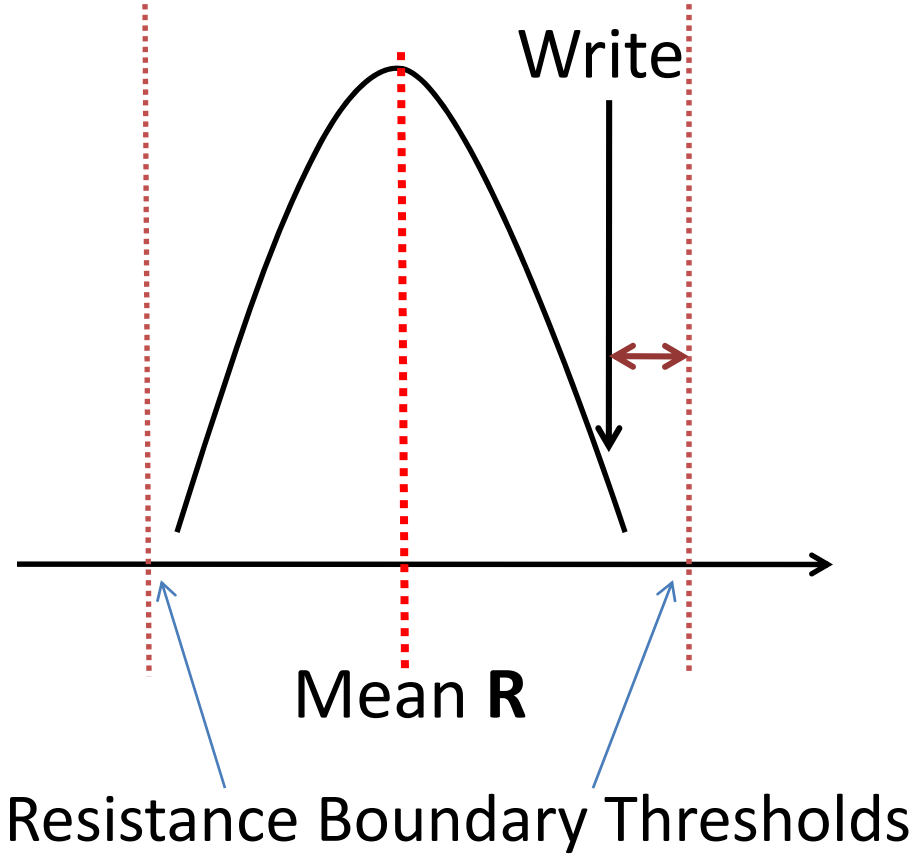
# Reducing Overheads with Circuit Level Solution

- Invoking ECC on every LARDD increases energy consumption
- ***Parity*** – like error detection circuit is used to signal the need for a full fledged ECC error detect
  - Number of **Drift Prone States** in each line are counted when the line is written into memory
  - **0** is stored as a ***Flag*** for even number of **Drift Prone States** , **1** for odd
  - The ***Flag*** is computed at each LARDD
  - A ***Flag*** mismatch invokes a full-fledged ECC
- Reduces need for ECC read-compare at every LARDD cycle



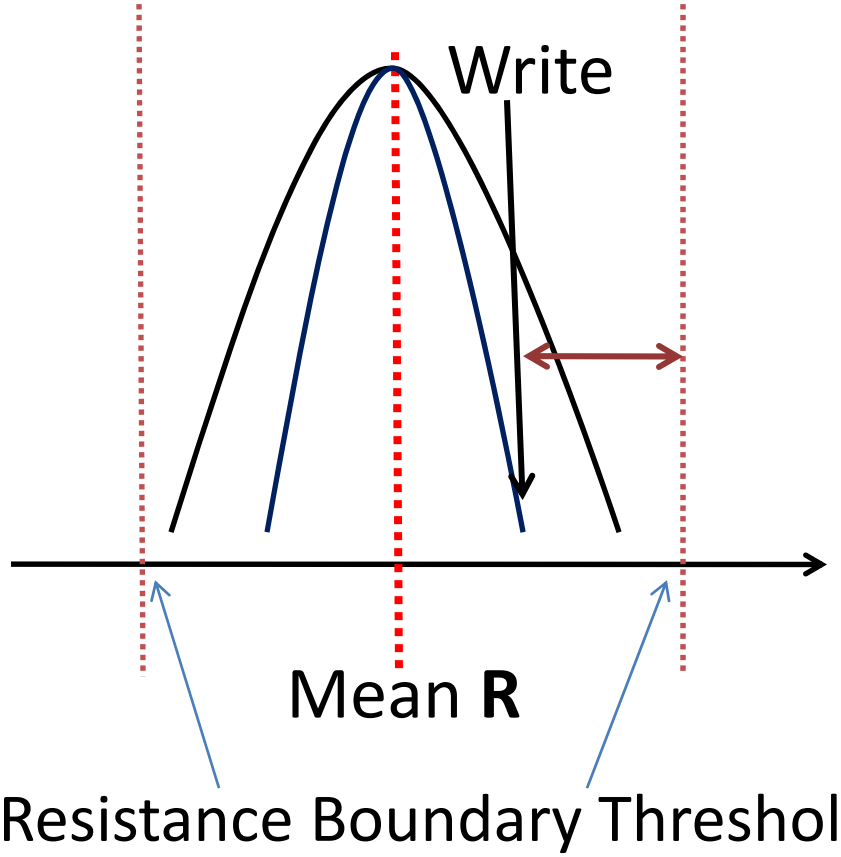

UTAH ARCH

# Device Level Solution – Precise Write




UTAH ARCH

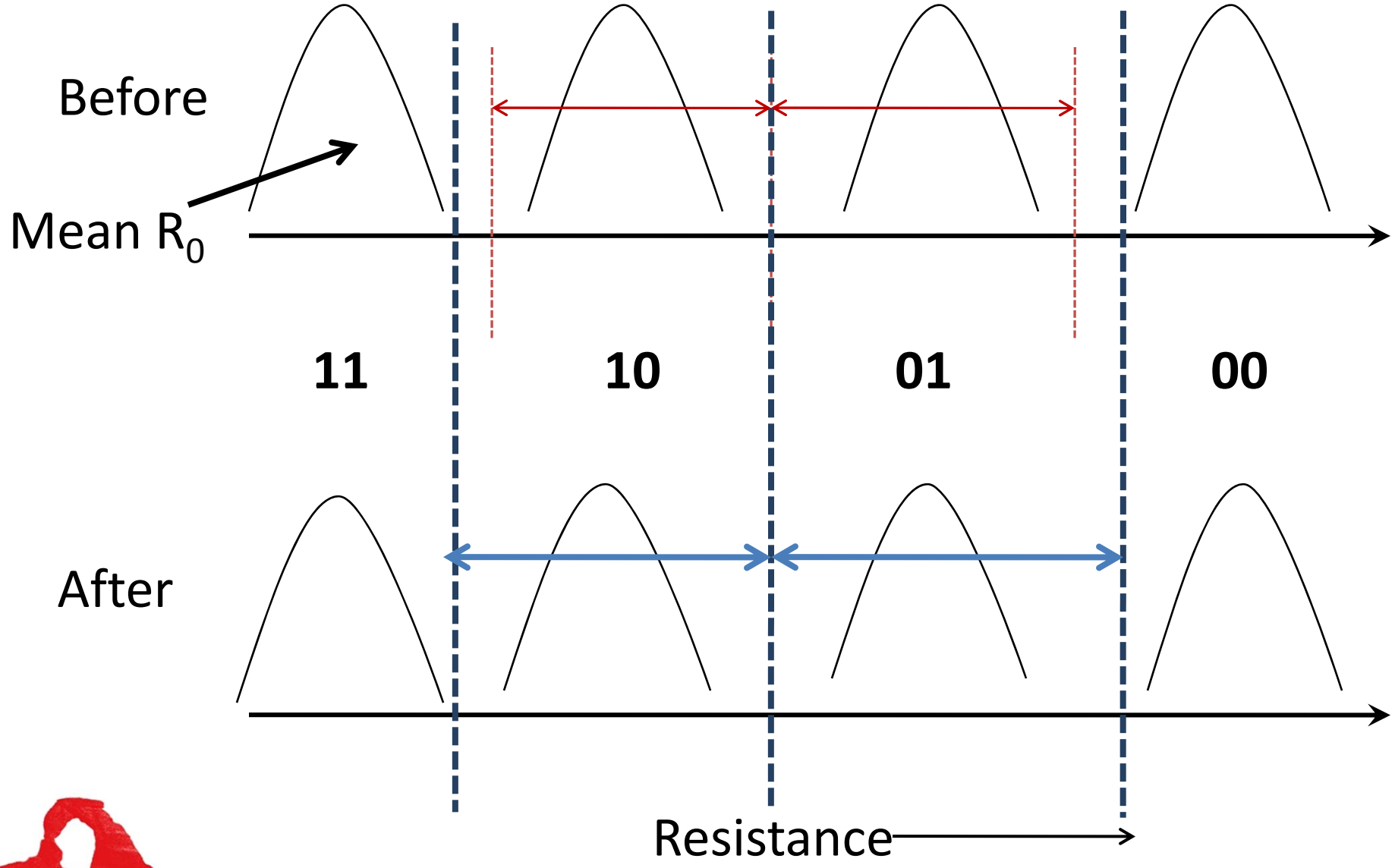
# Device Level Solution – Precise Write




Precise Writes help alleviate drift at device level but takes longer and hurts lifetime!

UTAH ARCH

# Device Level Solution – Non Uniform Banding




UTAH ARCH

# Solutions Summary

**Architectural**

- Hetero[illegible]geneities

Error rates vs lifetime

lifetime

**Device**

- G[illegible]

Error rates vs write energy

- [illegible] between error rate write energy

**Circuit**

- Parity has [illegible]ique

Reduces ECC overheads

- Red[illegible]overheads

**System**

- Vary[illegible]es

Accounts for varying conditions

[illegible]itions

[illegible]era[illegible]e

- Han[illegible] errors

UTAH ARCH

# Conclusions

- Resistance drift will exacerbate with MLC scaling
- Naïve solutions based on ECC support are costly for PCM
  - Increased write energy, decreased lifetimes
- Holistic solutions need to be explored to counter drift at device, architectural and system levels
  - 39% reduction in energy, 4x less errors, 102x increase in lifetime
  - Work in progress!!

UTAH ARCH

Thanks!!

www.cs.utah.edu/arch-research