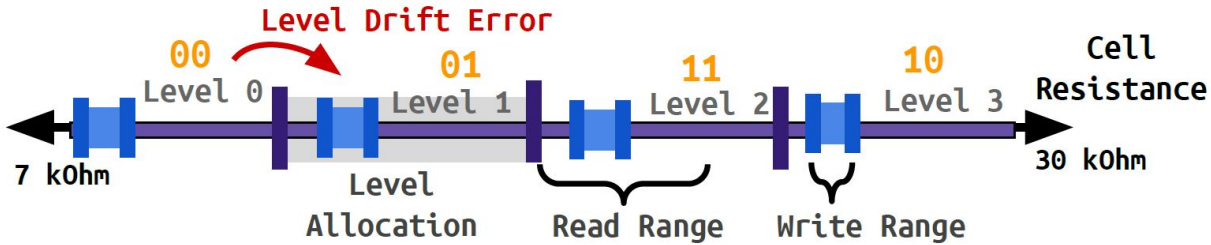
DALA: Distribution-Agnostic Level Allocation for Multiple Bits-Per-Cell RRAM

Anjiang Wei, Pu Yi, Akash Levy, Robert M. Radway, Priyanka Raina, Subhasish Mitra, Sara Achour

Background

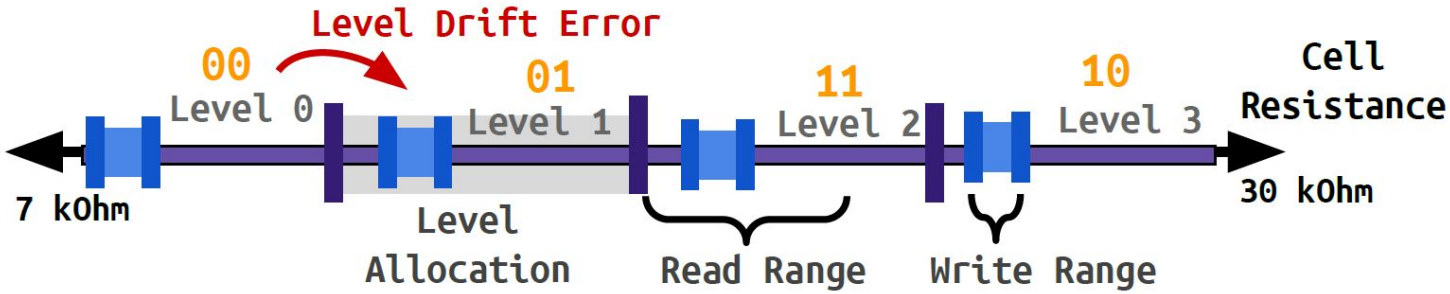
- Resistive random access memories (RRAM)
  - Analog storage
- Multiple-Bits-Per-Cell (MBPC)
  - 2 bits: 4 levels
  - 4 bits: 16 levels
- Mapping between levels and bits



An example 4-level (2 bit) allocation for resistive cell.

Background

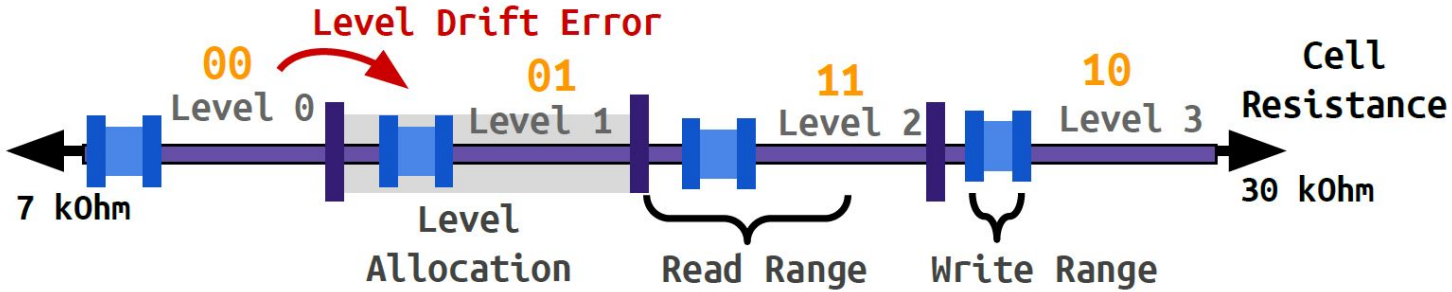
- Write Operation
- Read Operation
- Analog Non-idealities
  - Level Drift Errors
  - Bit Error Rate



An example 4-level (2 bit) allocation for resistive cell.

Background

- Level Allocation Algorithm
  - Write Range: Write Center, Write Tolerance
  - Read Range: Read Low, Read High



An example 4-level (2 bit) allocation for resistive cell.

Prior Work

- Level Allocation
  - Collect and characterize hardware data
  - Design algorithm based on characterization data
- Limitations of prior work (SBA)
  - Parameter-based hardware models
    - Standard deviation
    - Over-approximation (assuming normal distribution)
  - Non-Generalizable algorithm for level allocation
    - Parameter-based models

Contribution

- DALA: a distribution- and technology-agnostic level allocation
  - RDR resistance hardware model
  - More general, technology-agnostic algorithm
- RDR Resistance Model
  - **Distribution-agnostic** analog resistance model
- Level Allocation Algorithm
  - **No assumptions** about statistical trends in the data
  - **Device-agnostic search**, can potentially be applied to other technologies
- Comparison with prior work (SBA)
  - 29.6%-71.0% lower bit-error rates
  - 21.6%-40.9% lower ECC storage overheads

RDR Analog Resistance Model

- Two kinds of errors:
  - Write Error (write center **c**, write tolerance **z**)
  - Drift Error (Write distribution, time **t**)

$$\hat{r} \sim \mathbf{RDR}(c, z, t) = P_{wr}(c, z) + P_{dr}(P_{wr}(c, z), t)$$

- Operations
  - **PDF(RDR(c, z, t)):** construct a **probability density function** (PDF) from the model
  - **Percentile(P, F):** compute the **F percentile** from the probability density function
  - **Probability(P, r1, r2):** compute the **probability** the resistance is within [r1, r2]

Data Collection for RDR Model

- Write Resistance Dataset
  - Try different write centers **c** and write tolerances **z**
- Resistance Drift Dataset
  - Also try different timestamp **t**
- Construction of Distributions
  - Weighted linear interpolation over distributions
  - SBA: linear interpolation over standard deviation (sigma)

RDR Analog Resistance Model

- An abstraction layer, an interface between:
  - Real hardware data
  - Algorithm design
- Potentially transferable abstraction layer
  - For other hardware technologies
  - More data of the same hardware

Level Allocation Algorithm

- Basic Requirements
  - Read Ranges should NOT overlap
  - Write Ranges should be within Read Range
- Error evaluation:
  - Probability of writing to **level i**, ends up at **level j** at **time t**

$$e_{i,j}(t) = \mathbf{Probability}(\mathbf{PDF}(\mathbf{RDR}(c_i, z_i, t)), xl_j, xh_j)$$

  - Average drift error probability

$$e_{avg}(t) = \frac{1}{n}\sum_{i=1}^{n} 1 - e_{i,i}(t)$$

Level Allocation Algorithm

- Basic Algorithm
- Input:
  - RDR:a hardware model
  - n: number of levels
  - t: timestamp
  - error margin
  - C, Z: defining the search space (write centers, write tolerances)
- Idea: progressively increases a maximum level drift error by the provided margin until it finds a satisfying level allocation of n levels

Function `LevelAlloc` $(RDR, n, t, \epsilon, C, Z)$ **:**
  for $\gamma \in [0, \epsilon, 2\epsilon, ..., 1]$ **do**
    $Levels =$ `LevelGeneration` (RDR, t, γ, C, Z)
    $L =$ `FindAllocation` $(Levels)$
    if $len(L) == n$ **then**
      return L

Level Allocation Algorithm

- Search Space Construction
- Idea: construct a candidate of levels w.r.t. the error bound specification
  - These candidates may overlap with each other
  - We need to pick longest non-overlapping levels for read ranges

Function `LevelGeneration` (RDR, t, γ, C, Z)**:**

  for $\langle c, z \rangle$ *in* $C \times Z$ **do**

    P = `PDF`(`RDR`(c, z, t))

    xl = `Percentile`$(P, [\frac{1}{2} \cdot \gamma] \times 100\%)$

    xh = `Percentile`$(P, [1 - \frac{1}{2} \cdot \gamma] \times 100\%)$

    assert $1 -$ `Probability`$(P, xl, xh) \leq \gamma$

    assert $[c - z, c + z] \subseteq [xl, xh]$

    generate $\langle xl, xh, c, z \rangle$

Level Allocation Algorithm

- Level Allocation Construction
- Idea: find a level allocation with the largest number of non-overlapping levels

Function `FindAllocation` $(Levels)$**:**

$LevelAlloc = []$ # Selected Non-Overlapping Levels

$SortLevels$ = `sort`$(Levels, key = xh)$

$LargestR = 0$

for $\langle xl, xh, c, z\rangle$ *in* $SortLevels$ **do**

if $|LevelAlloc| == 0 \vee xl \geq LargestR$ **then**

$LargestR = xh$

$LevelAlloc = LevelAlloc :: \langle xl, xh, c, z\rangle$

return $LevelAlloc$

Level Allocation Algorithm

- We can guarantee that algorithm will always find an n-level allocation if one exists in the search space
- Algorithm:
  - First sorts all the generated candidate levels by their upper read resistance xh
  - Iteratively adds the leftmost candidate level that does not overlap with the rightmost level in the level allocation

Function `FindAllocation` $(Levels)$ **:**

$LevelAlloc = []$ # Selected Non-Overlapping Levels

$SortLevels$ = `sort`$(Levels, key = xh)$

$LargestR = 0$

for $\langle xl, xh, c, z\rangle$ *in* $SortLevels$ **do**

if $|LevelAlloc| == 0 \vee xl \geq LargestR$ **then**

$LargestR = xh$

$LevelAlloc = LevelAlloc :: \langle xl, xh, c, z\rangle$

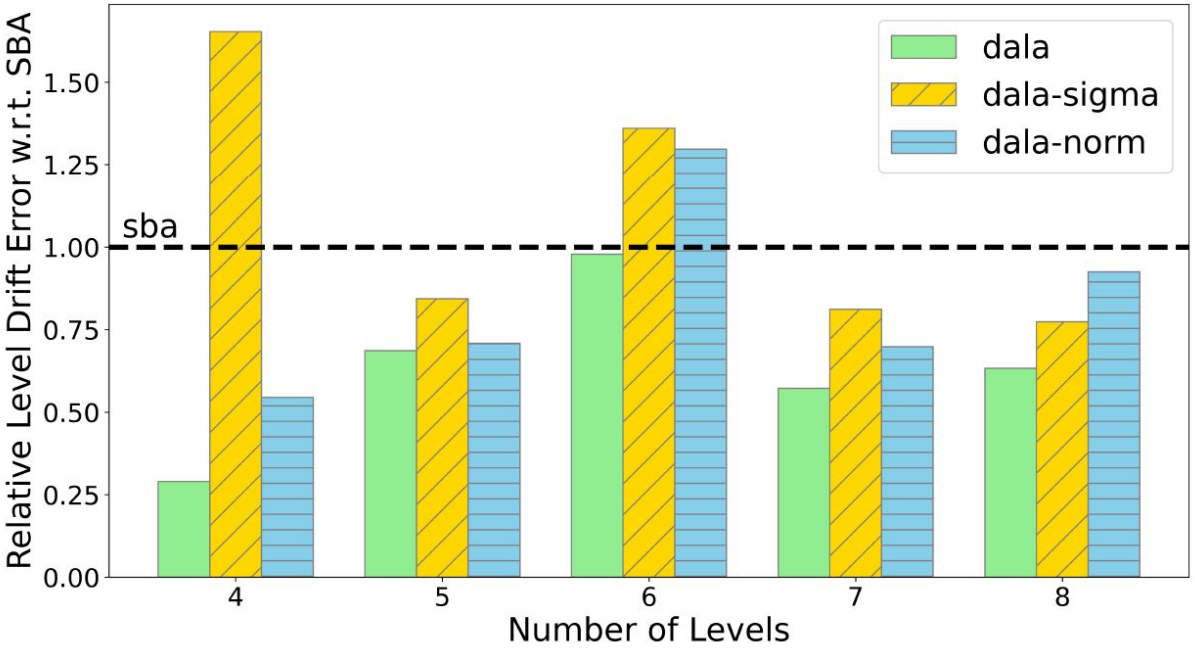
return $LevelAlloc$

Experimental Setup

- Baselines for comparison
  - SBA
  - Dala-sigma: same hardware model as SBA
  - Dala-norm: assuming normal distribution
- Data collection
  - All baselines share the same dataset

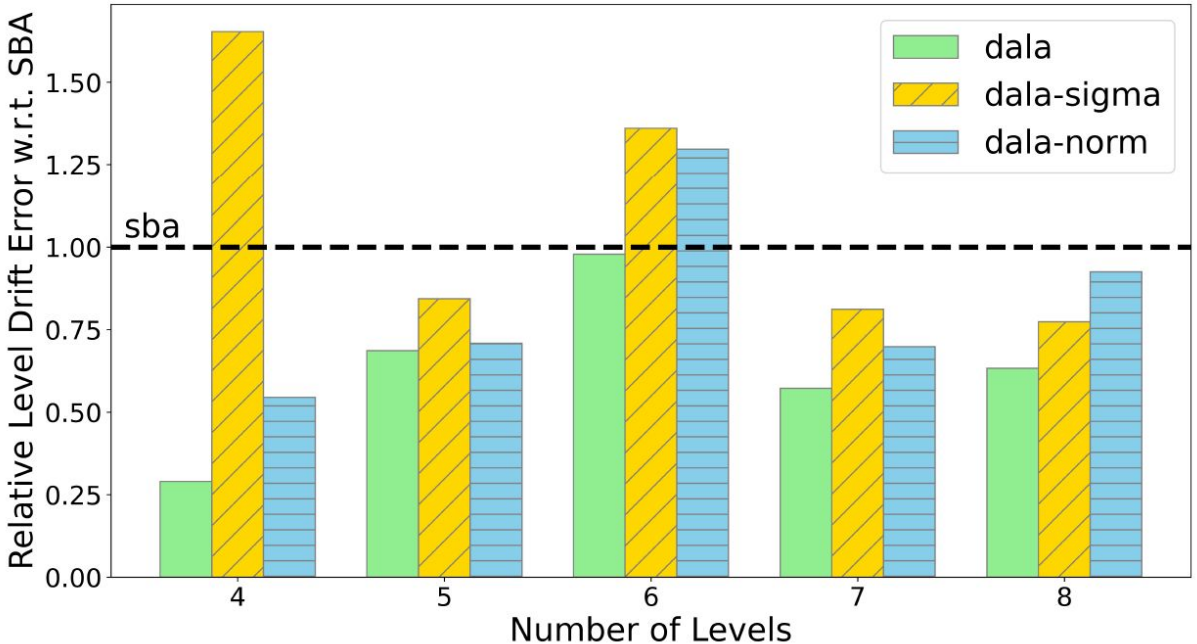
Result Analysis

- DALA consistently produces lower error level allocations than SBA



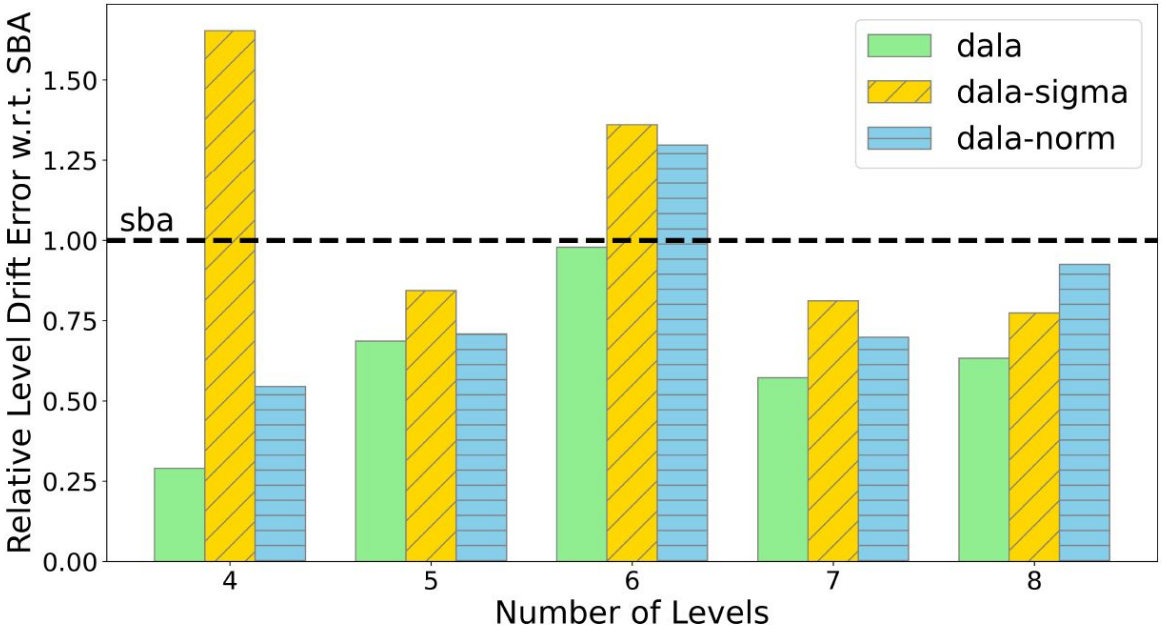
Result Analysis

- The **RDR model** enables DALA to produce much lower-error allocations
- Compare dala & dala-sigma



Result Analysis

- Assuming normal distribution leads to sub-optimal allocation
- Compare dala & dala-norm



Result Analysis

- Almost all write/relaxation data indicate that the distribution is NOT normal
  - Radar: write dataset
  - Tech A/B/C: relaxation dataset

			Normal Percentage		
dataset	**number of cells**	**measurement**	**write**	**relax**	**read**
DALA	200	resist.	2.1%	0%	0%
DALA	200	cond.	2.1%	0%	0%
radar	8192	resist.	0%	-	-
radar	8192	cond.	0%	-	-
tech A	16384	resist.	-	8.2%	-
tech A	16384	cond.	-	8.7%	-
tech B	32768	resist.	-	4.5%	-
tech B	32768	cond.	-	6.6%	-
tech C	16292	resist.	-	0.6%	-
tech C	16292	cond.	-	3.6%	-

Table I

D'AGOSTINO'S K-SQUARED NORMALITY TEST RESULTS FOR RRAM CHARACTERIZATION DATASETS.

Experiment Results

- Compared to SBA
- Bit Error Rates
  - 71.0% BER reduction for 2 bits-per-cell (BPC)
  - 29.6% BER reduction for 3 BPC
- Error Correcting Code Overheads
  - 40.9% reduction in ECC overhead for 2 BPC
  - 21.6% reduction in ECC overhead for 3 BPC

Conclusion

- DALA: a distribution- and technology-agnostic level allocation
  - RDR resistance hardware model
  - More general, technology-agnostic algorithm
- Prior work (SBA)
  - Fit a known distribution to the observed analog behavior
  - Exploit device-specific statistical trends
- Comparison with SBA
  - 29.6%-71.0% lower bit-error rates
  - 21.6%-40.9% lower ECC storage overheads