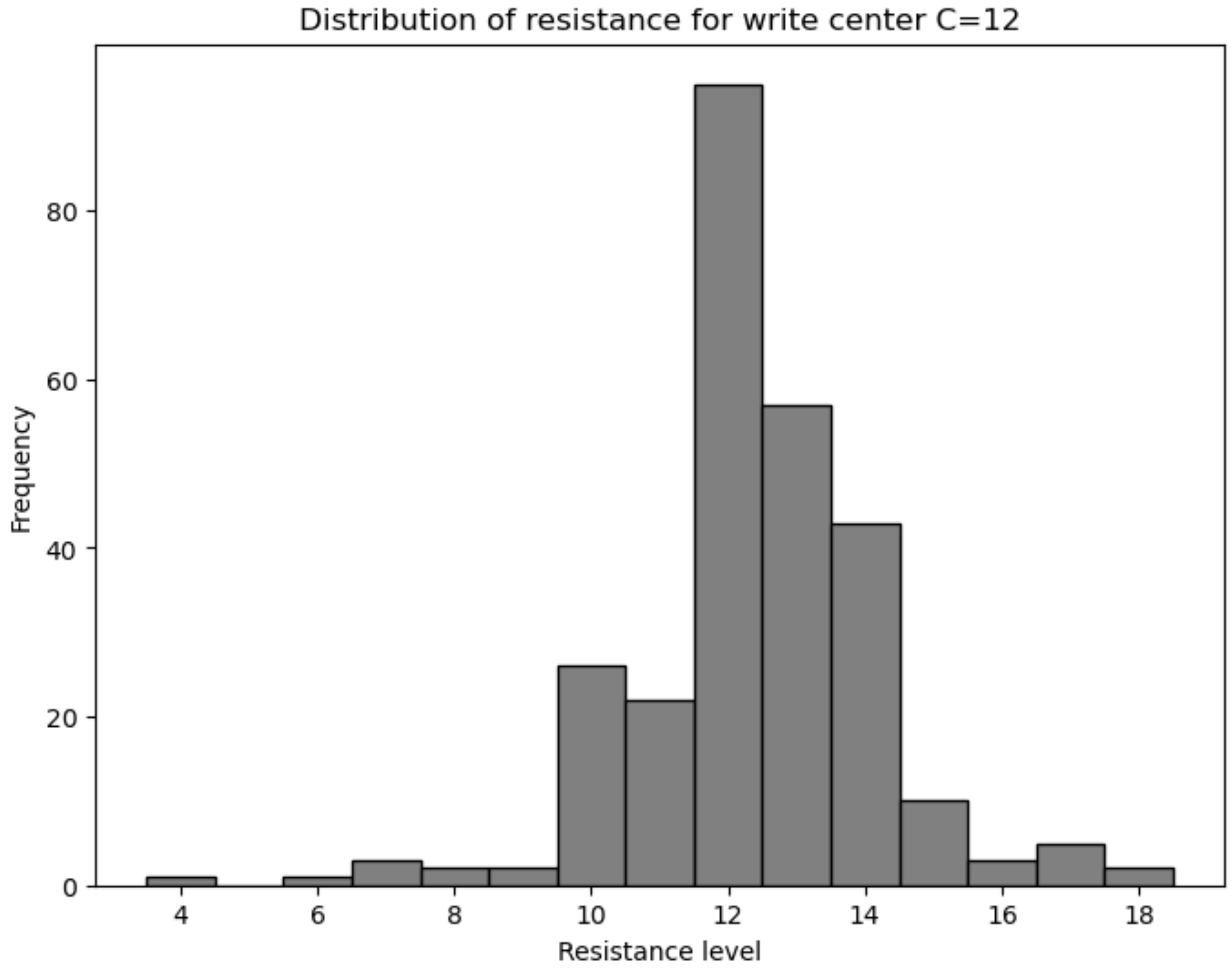
FPBA: **F**lexible **P**ercentile-**B**ased **A**llocation for Multiple-Bits-Per-Cell RRAM

Junfei Liu and Anson Kahng
University of Rochester

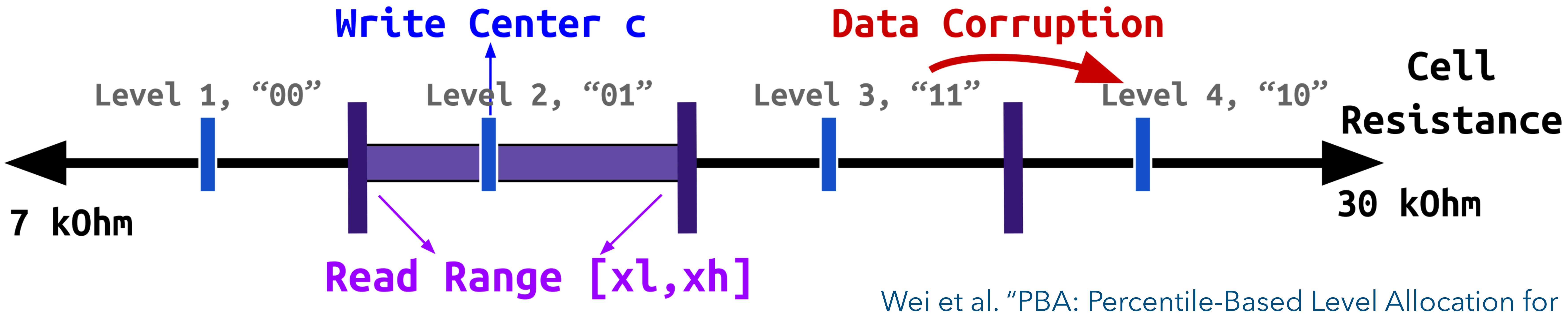
Resistive Random Access Memory (RRAM)

- Non-volatile memory technology
- Stores data by changing resistance with voltage
- Resistance value set at write operation ("write center")
- Smaller voltage used to measure the resistance multiple times to characterize the resistance distribution
- Allows for multiple-bits-per-cell (MBPC)



Multiple-Bit-Per-Cell (MBPC) Level Allocation

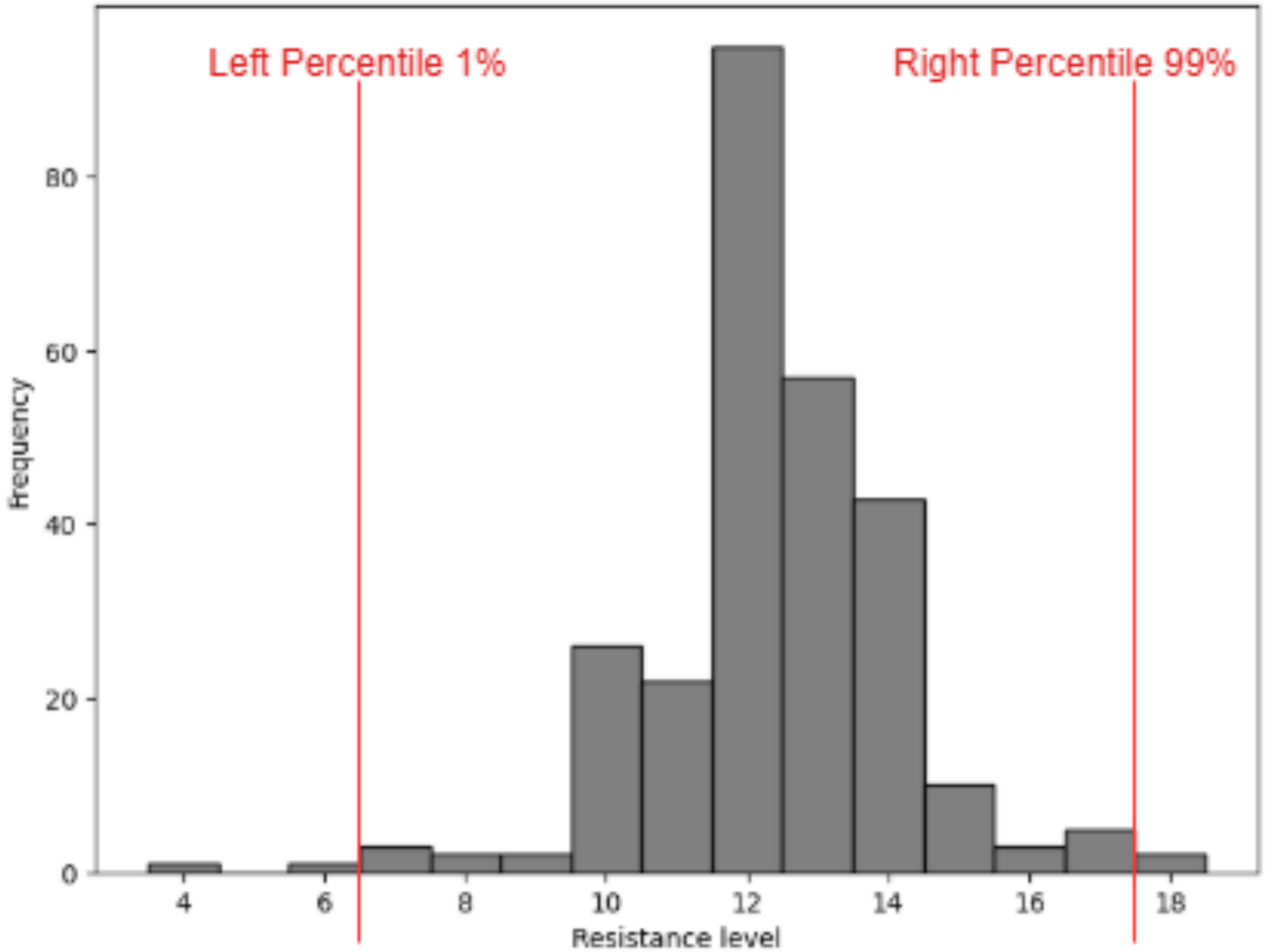
- Central question:
  How to partition MBPC RRAM cells into non-overlapping levels with low error?
- Level allocation algorithm:
  - Map bit combinations to resistance ranges
  - Write center c, read range $[xl, xh]$
  - *Data corruption*: Write to level 3 (“11”), read from level 4 (“10”) → one bit flip



Wei et al. “PBA: Percentile-Based Level Allocation for Multiple-Bits-Per-Cell RRAM”. ICCAD 2023.

Core Concepts

- Error probability (γ): Maximum allowable probability of a bit error of an allocated level
- Example: γ = 2% with a level $xl = 7$ (inclusive), $xl = 18$ (exclusive)



Core Concepts

- *Gray coding*: Encoding that ensures one bit flip between adjacent levels
- *Bit error rate (BER)*:

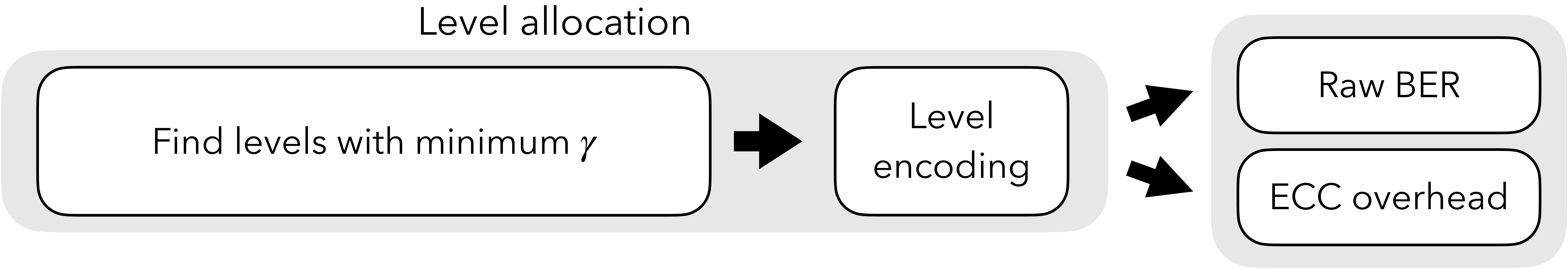
$$\frac{\text{Number of bit flips}}{\text{Total bits}} \times 100\,\%$$

- *Error-correcting code overhead (ECC)*: Fraction of additional bits needed to protect against errors (Reed-Solomon, BCH, or Hamming encoding) – related to BER

Resistance state	**Gray-coded bit value**
R1 (highest)	"00"
R2	"01"
R3	"11"
R4 (lowest)	"10"

Objectives

- Minimize γ during level allocation
- Minimize BER and ECC overhead in the end
- Overall flow:



Related Literature

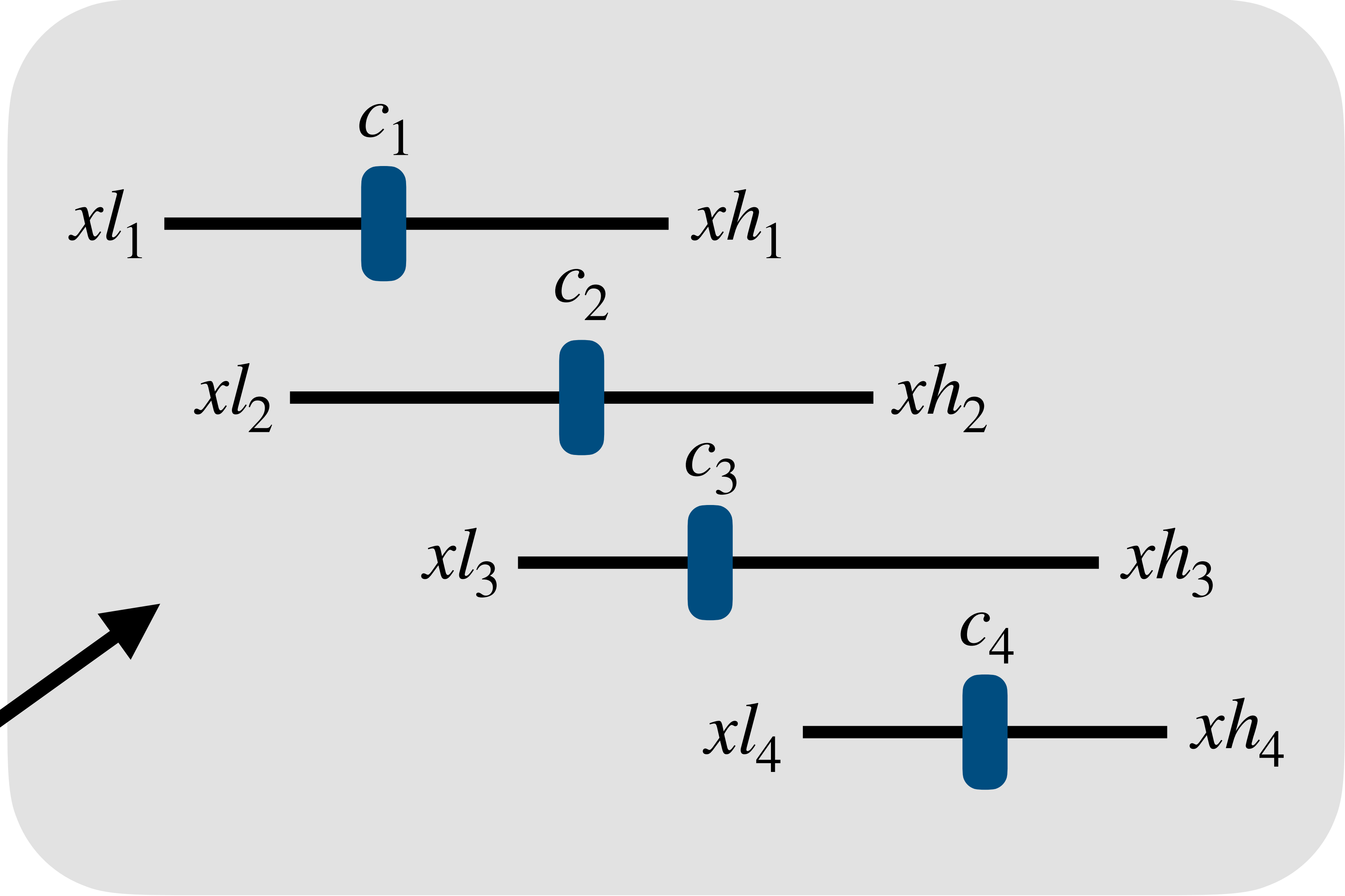
- Sigma-Based Allocation (SBA)[1]:
  - Fit distributions to characterization data
  - Parameterization not always applicable
- Percentile-Based Allocation (PBA)[2]:
  - State-of-the-art
  - Directly work with characterization data
  - Capture analog behaviors present in the data
  - Great improvement over SBA

[1]: Le et al. “RADAR: A fast and energy-efficient programming technique for multiple bits-per-cell RRAM arrays”. IEEE Transactions on Electron Devices. 2021.

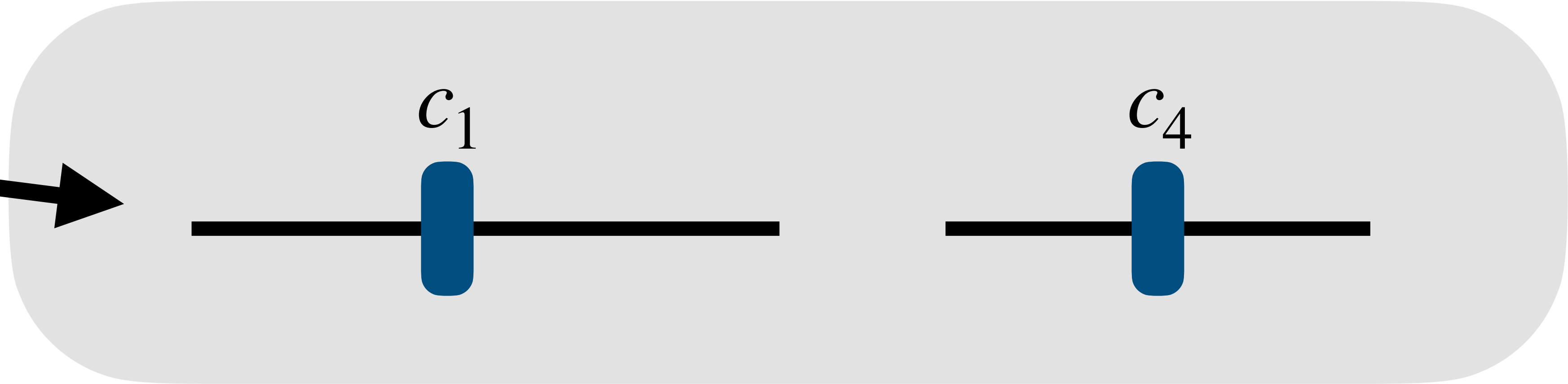
[2]: Wei et al. “PBA: Percentile-Based Level Allocation for Multiple-Bits-Per-Cell RRAM”. ICCAD 2023.

Percentile-Based Allocation (PBA)

- Level Allocation (LA) subroutine:
  - **Input**: n (target number of levels in cell) and characterization dataset
  - **Goal**: Find minimum γ (error probability)
  - Get candidate levels: $[\gamma/2,\ 1-\gamma/2]$ (cut off ends symmetrically)



  - Sort and find first n non-overlapping candidate levels

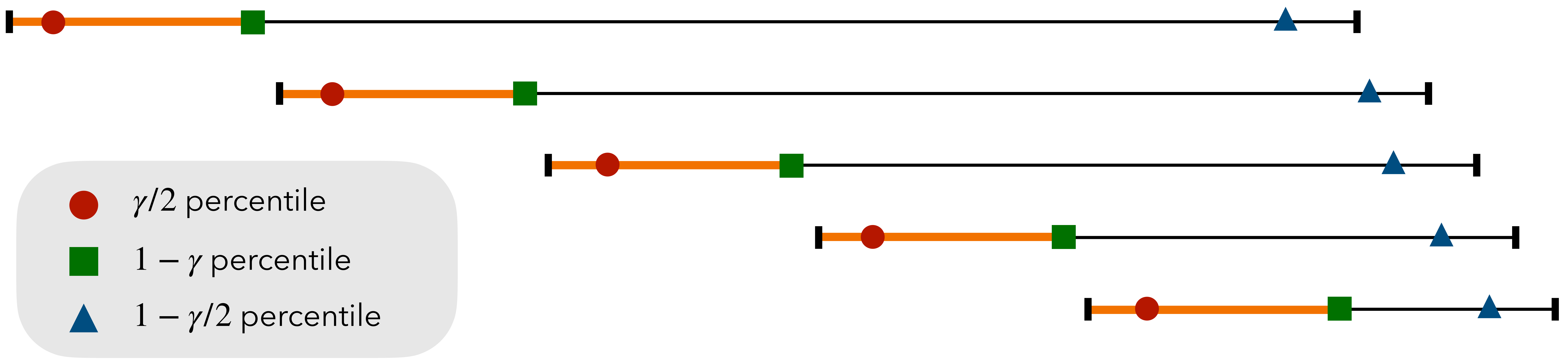


  - If a solution exists, return

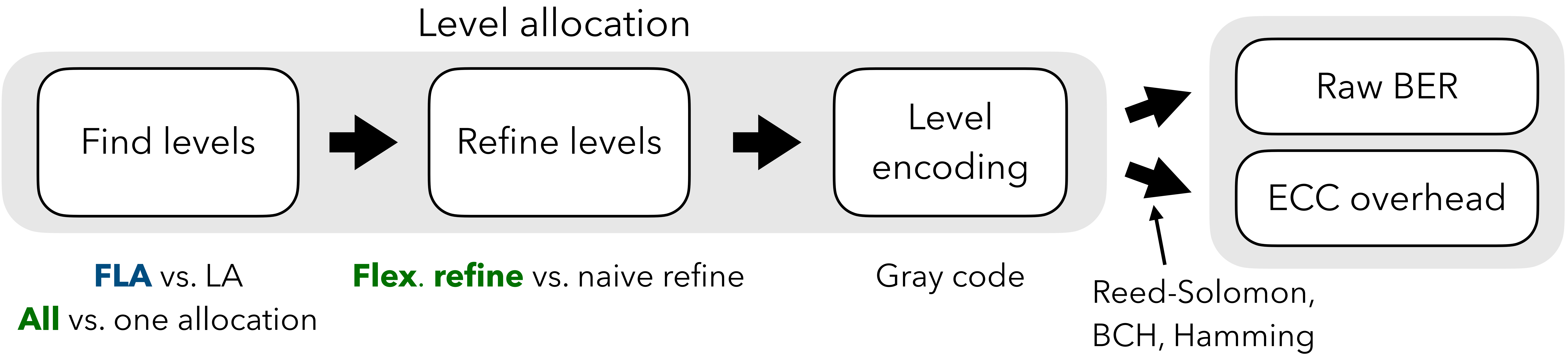
Wei et al. "PBA: Percentile-Based Level Allocation for Multiple-Bits-Per-Cell RRAM". ICCAD 2023.

PBA Limitations

- **Theorem** (informal): For any error probability γ, target number of levels n, and number of write centers $c \geq n$, in the worst case, LA finds an arbitrarily poor approximation of the optimal level allocation
- "Proof by picture":

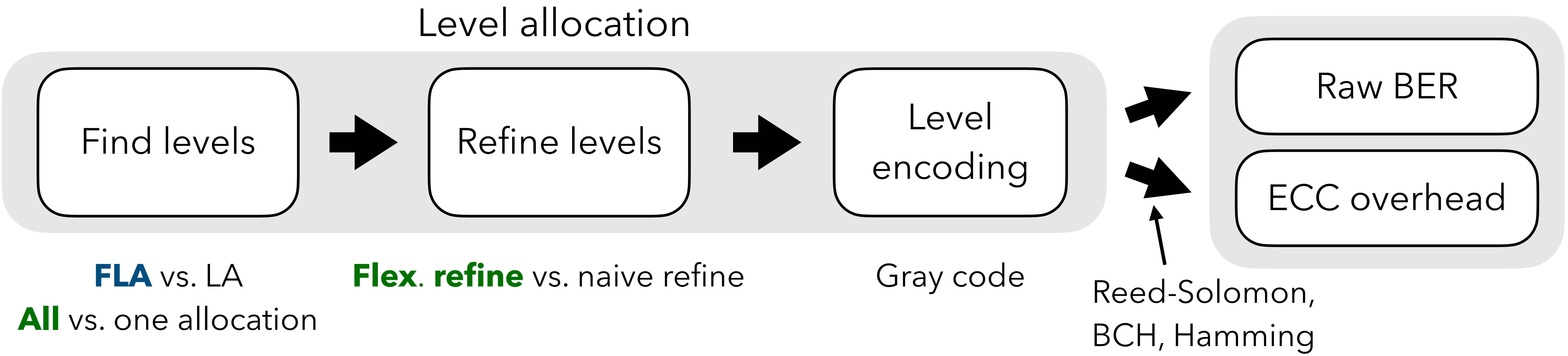


FPBA: **F**lexible **P**ercentile-**B**ased **A**llocation



- **Flexible level allocation (FLA)**: finds *provably optimal* γ
- Heuristic optimizations:
  - **Find all cliques (AC)**, **flexible level refinement (FR)**

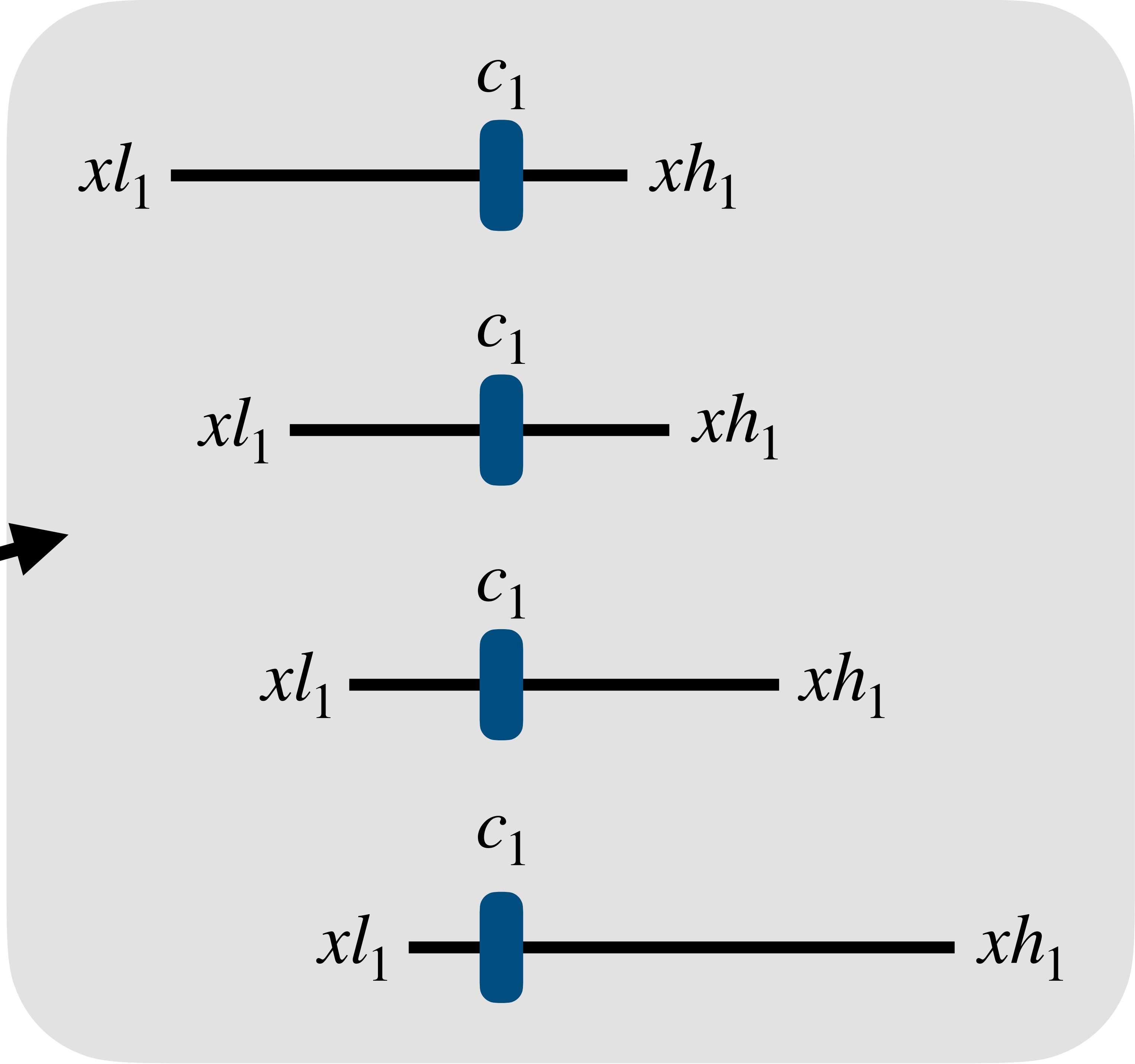
FPBA: **F**lexible **P**ercentile-**B**ased **A**llocation



- **Flexible level allocation (FLA)**: finds *provably optimal* γ
- Heuristic optimizations:
  - **Find all cliques (AC)**, **flexible level refinement (FR)**

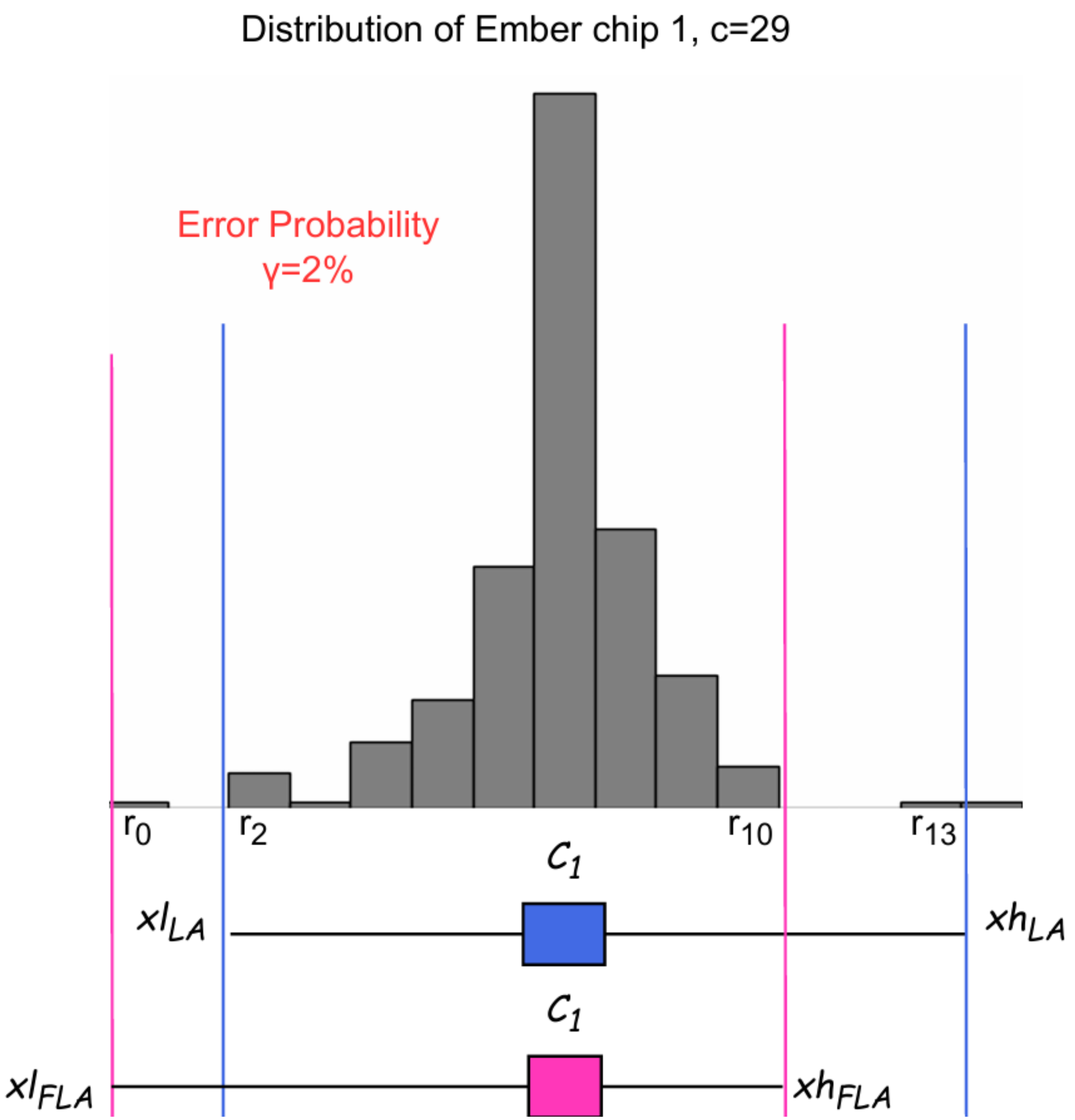
FPBA: Flexible Level Allocation (FLA)

- **Input**: n (target number of levels in cell) and characterization dataset
- **Goal**: Find minimum γ (error probability)
- Get candidate levels: $[0,\ 1-\gamma]$ through $[\gamma, 1]$ (cut off ends asymmetrically)
- Sort and find first n non-overlapping candidate levels and update candidate levels' ranges
- If a solution exists, return



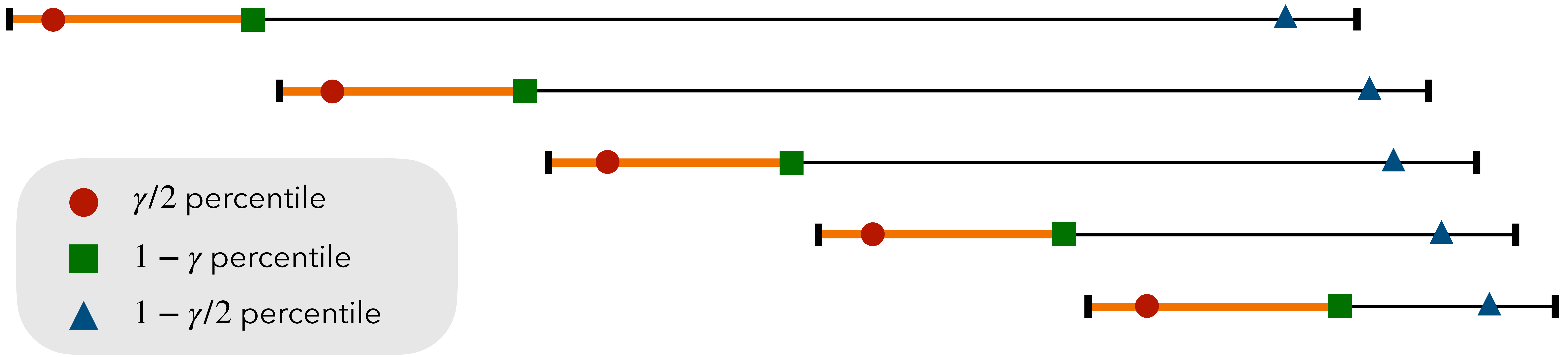
Flexible Level Allocation (FLA) vs. Level Allocation (LA)

- FLA strictly generalizes LA
- Sometimes leads to better performance



Flexible Level Allocation (FLA) Optimality

- **Theorem** (informal): For any error probability γ and input characterization dataset, FLA returns an allocation with the optimal number of levels
- Proof idea: "Greedy stays ahead"
- Example:



γ vs. BER: Error Probability ≠ Bit Flips

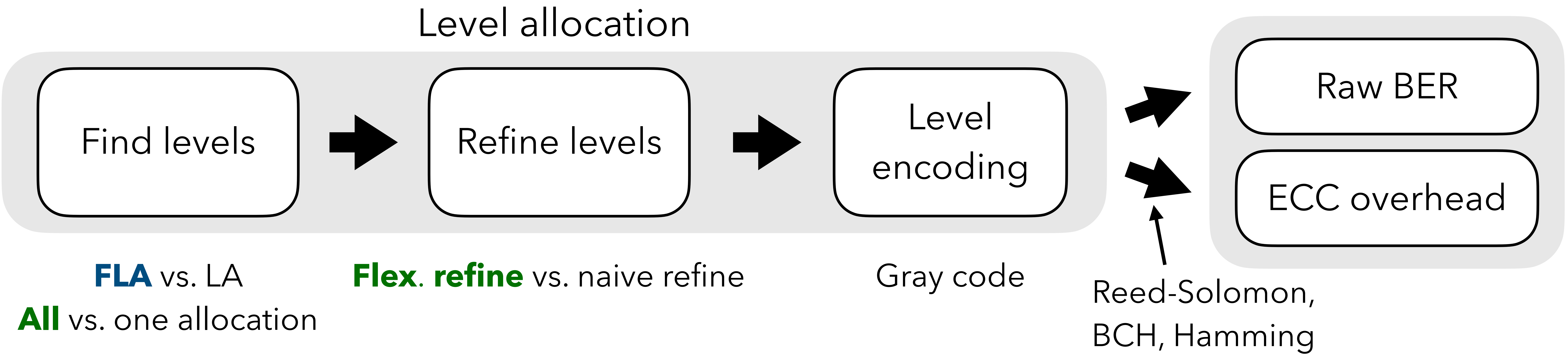
- γ: maximum error probability
- Bit error rate (BER): penalize errors by number of bit flips between levels

$$BER = \frac{\text{Number of bit flips}}{\text{Total bits}} \times 100\,\%$$

- BER is bounded by γ, yet smaller γ does not necessarily mean smaller BER

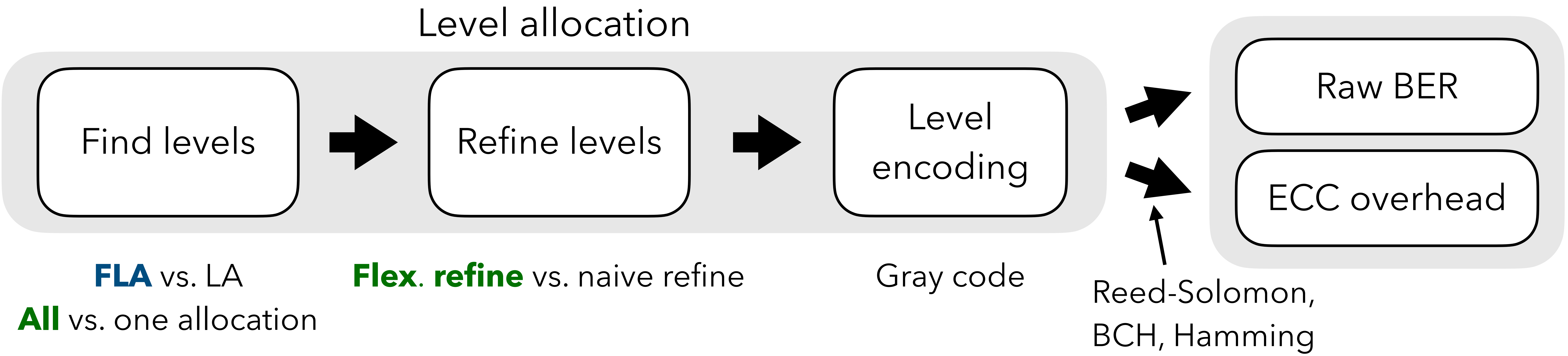
$$\frac{\gamma}{\lceil \log_2(n) \rceil} \leq BER \leq \gamma$$

FPBA: **F**lexible **P**ercentile-**B**ased **A**llocation



- **Flexible level allocation (FLA)**: finds *provably optimal* γ
- Heuristic optimizations:
  - **Find all cliques (AC)**, **flexible level refinement (FR)**

FPBA: **F**lexible **P**ercentile-**B**ased **A**llocation



- **Flexible level allocation (FLA)**: finds *provably optimal* γ

- Heuristic optimizations:
  - **Find all cliques (AC)**, **flexible level refinement (FR)**

FPBA Heuristic #1: Flexible Refinement (FR)

- An allocation that satisfies a minimum error probability γ may have gaps
- Question: *How should we distribute elements in the gaps to optimize BER and ECC?*
- **Flexible refinement (FR)**: Try all possible distributions

FPBA Heuristic #2: Find All Cliques (AC)

- Both LA and FLA find the lexicographically *first* allocation satisfying the minimum γ requirement
- Insight: Sometimes it's better to find a different allocation! Being first likely means that the solution has some "skew"
- **Find all cliques (AC)**: Choose among different level allocations that all achieve the same γ to optimize BER and ECC
  - Define an equivalent graph to the level allocation problem (vertices = candidate levels given γ, edges = non-overlap between levels)
  - Find all admissible level allocations == find all cliques

Experiments

- Evaluate FPBA (FLA + FR + AC) on two fabricated RRAM storage arrays
  - EMBER cells with 64 resistance levels and write centers

Chip	# Total Cells	Readout	Resistance	# Tested Write Centers	# Test Cells
Ember 1	3M	On-chip ADC	1 - 64 levels	64	16K
Ember 2	3M	On-chip ADC	1 - 64 levels	64	16K

Upton et al. "EMBER: a 100 MHz, 0.86mm2, Multiple-Bits-per-Cell RRAM Macro in 40 nm CMOS with Compact Peripherals and 1.0 pJ/Bit Read Circuitry," ESSCIRC 2023

Experimental Setup

- Baseline: PBA (LA on its own)
- Metrics:
  - Error probability (γ)
  - Bit error rate (BER)
  - Error-correcting code overhead (ECC)
- Allocations: 8-level (3 bits-per-cell) and 16-level (4 bits-per-cell)
  - PBA and FPBA both do perfectly on 4-level (2 bits-per-cell) allocations

Results: FLA vs. LA

- Reductions in γ: over 30% for 3 bits-per-cell, over 27% for 4 bits-per-cell
- However, this results in BER reductions only for 3 bits-per-cell, and BER increases for 4 bits-per-cell (γ is only an upper bound on BER)

Chip	bpc	Max Error Prob γ				Bit Error Rate (BER)				ECC Overhead			
		LA	FLA	$\Delta\gamma$	% $\Delta\gamma$	LA	FLA	ΔBER	% ΔBER	LA	FLA	ΔECC	% ΔECC
Ember1	3	2.2%	**1.6%**	-0.68%	-30%	0.38%	**0.35%**	-0.03%	-7.8%	**9.1%**	**9.1%**	0%	0%
Ember2	3	3.0%	**1.9%**	-1.2%	-39%	0.37%	**0.35%**	-0.015%	-4.2%	**9%**	**9%**	0%	0%
Ember1	4	26%	**19%**	-7.0%	-27%	**3.6%**	3.7%	0.015%	0.4%	**32%**	**32%**	0%	0%
Ember2	4	30%	**21%**	-9.2%	-30%	**3.7%**	4.0%	0.36%	9.9%	**32%**	**32%**	0%	0%

Results: FLA vs. LA

- Reductions in γ: over 30% for 3 bits-per-cell, over 27% for 4 bits-per-cell
- However, this results in BER reductions only for 3 bits-per-cell, and BER increases for 4 bits-per-cell (γ is only an upper bound on BER)

Chip	bpc	Max Error Prob γ				Bit Error Rate (BER)				ECC Overhead			
		LA	FLA	$\Delta\gamma$	% $\Delta\gamma$	LA	FLA	ΔBER	% ΔBER	LA	FLA	ΔECC	% ΔECC
Ember1	3	2.2%	**1.6%**	-0.68%	-30%	0.38%	**0.35%**	-0.03%	-7.8%	**9.1%**	**9.1%**	0%	0%
Ember2	3	3.0%	**1.9%**	-1.2%	-39%	0.37%	**0.35%**	-0.015%	-4.2%	**9%**	**9%**	0%	0%
Ember1	4	26%	**19%**	-7.0%	-27%	**3.6%**	3.7%	0.015%	0.4%	**32%**	**32%**	0%	0%
Ember2	4	30%	**21%**	-9.2%	-30%	**3.7%**	4.0%	0.36%	9.9%	**32%**	**32%**	0%	0%

Results: FLA + AC vs. LA

- Improvements in all cases
- Exponential complexity
- Fewer possible level allocations at the minimum possible γ at 4 bits-per-cell, therefore smaller room for improvements

Chip	bpc	Bit Error Rate (BER)						ECC Overhead					
		LA	LA+AC	% ΔBER	FLA	FLA+AC	% ΔBER	LA	LA+AC	% ΔECC	FLA	FLA+AC	% ΔECC
Ember1	3	0.38%	**0.29%**	-24%	0.35%	**0.29%**	-18%	9.1%	**8.1%**	-11%	9.1%	**8.1%**	-11%
Ember2	3	0.37%	0.26%	-29%	0.35%	**0.25%**	-30%	9.0%	7.7%	-15%	9.0%	**7.6%**	-16%
Ember1	4	3.6%	3.5%	-1.2%	3.7%	-	-	32%	31%	-1.4%	32%	-	-
Ember2	4	3.7%	3.5%	-3.1%	4.0%	-	-	32%	31%	-2.1%	32%	-	-

Results: FLA + AC vs. LA

- Improvements in all cases
- Exponential complexity
- Fewer possible level allocations at the minimum possible γ at 4 bits-per-cell, therefore smaller room for improvements

Chip	bpc	Bit Error Rate (BER)						ECC Overhead					
		LA	LA+AC	% ΔBER	FLA	FLA+AC	% ΔBER	LA	LA+AC	% ΔECC	FLA	FLA+AC	% ΔECC
Ember1	3	0.38%	**0.29%**	-24%	0.35%	**0.29%**	-18%	9.1%	**8.1%**	-11%	9.1%	**8.1%**	-11%
Ember2	3	0.37%	0.26%	-29%	0.35%	**0.25%**	-30%	9.0%	7.7%	-15%	9.0%	**7.6%**	-16%
Ember1	4	3.6%	3.5%	-1.2%	3.7%	-	-	32%	31%	-1.4%	32%	-	-
Ember2	4	3.7%	3.5%	-3.1%	4.0%	-	-	32%	31%	-2.1%	32%	-	-

Results: FLA + AC + FR vs. LA

- Best performance: entire pipeline of theoretical and empirical optimizations
- Format: BER, ECC overhead

Method	Ember1 (3 bpc)	Ember2 (3 bpc)	Ember1 (4 bpc)	Ember2 (4 bpc)
LA	0.38%, 9.1%	0.37%, 9.0%	3.6%, 32%	3.7%, 32%
LA+FR	0.38%, 9.1%	0.34%, 9.0%	**3.5%**, 32%	3.7%, 32%
FLA	0.35%, 9.1%	0.35%, 9.0%	3.7%, 32%	4.0%, 32%
FLA+FR	0.33%, 9.1%	0.34%, 9.0%	3.7%, 32%	3.9%, 32%
LA+AC	**0.29%, 8.1%**	0.26%, 7.7%	3.6%, 31%	**3.5%, 31%**
LA+AC+FR	**0.29%, 8.1%**	**0.25%, 7.6%**	**3.5%, 30%**	**3.5%, 31%**
FLA+AC	**0.29%, 8.1%**	**0.25%**, 7.7%	–	–
FLA+AC+FR	**0.29%, 8.1%**	**0.25%**, 7.7%	–	–

Takeaways

- FLA produces provably optimal γ
- Heuristic steps (AC, FR) meaningfully optimize toward optimal BER / ECC
- Empirical results at a glance:
  - 27 - 39% lower γ
  - 2.8 - 32.4% lower BER
  - 3.1 - 15.6% lower ECC overhead

Limitations and Future Work

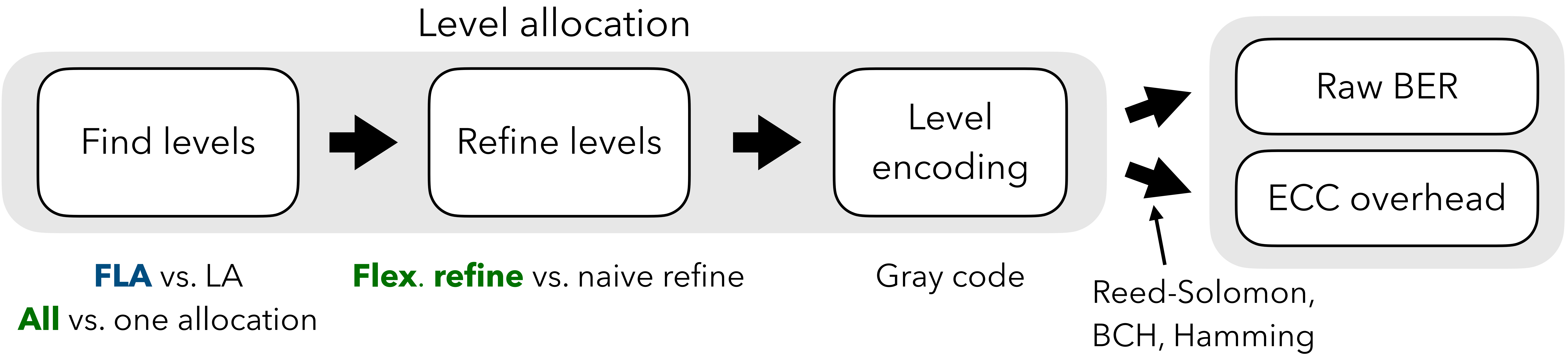
- Limitations:
  - Find all cliques (AC) is prohibitively computationally expensive
  - High dependency on characterization data
- Future Work:
  - Find all cliques (AC): approximation / sampling
  - Relax γ during BER / ECC optimization
    - Initial experiments up to 300% γ: BER increases as γ increases
  - Go beyond Gray coding (tailor coding scheme to be purely data-driven?)

Acknowledgments

- Anjiang Wei and Akash Levy: helpful correspondence about PBA
- Andrew Kahng: bringing this problem to our attention

Questions

- Thank you!



Appendix: Partial Dataset

Size	Ember1 Bit Error Rate			
	LA	FLA	LA+AC	FLA+AC
25%	0.58 ± 0.16	0.58 ± 0.16	**0.57** ±0.17	**0.57** ±0.15
50%	0.52 ± 0.10	0.47 ± 0.07	0.41 ± 0.05	**0.39** ±0.05
75%	0.42 ± 0.07	0.41 ± 0.04	0.34 ± 0.04	**0.32** ±0.04
90%	0.38 ± 0.04	0.37 ± 0.04	**0.29** ±0.01	**0.29**±0.01
100%	0.38	0.35	**0.29**	**0.29**

Size	Ember2 Bit Error Rate			
	LA	FLA	LA+AC	FLA+AC
25%	0.5 ± 0.12	0.5 ± 0.12	**0.47** ±0.11	0.55 ± 0.12
50%	0.58 ± 0.09	0.53 ± 0.08	**0.36** ±0.09	0.41 ± 0.09
75%	0.43 ± 0.06	0.46 ± 0.06	0.29 ± 0.04	**0.27** ±0.03
90%	0.46 ± 0.10	0.38 ± 0.06	**0.26** ±0.03	0.27 ± 0.04
100%	0.37	0.35	0.26	**0.25**

Appendix: Interchip Dataset

Mix	Ember1 Bit Error Rate							
	LA	LA+FR	FLA	FLA+FR	LA+AC	LA+AC+FR	FLA+AC	FLA+AC+FR
100/0	0.38	0.38	0.35	0.33	**0.29**	**0.29**	**0.29**	**0.29**
50/50	0.46	0.39	0.56	0.51	0.46	**0.38**	0.44	0.41
10/90	0.65	**0.63**	0.78	0.71	0.69	0.67	0.67	0.67
0/100	**0.64**	**0.64**	0.71	0.71	0.69	0.67	0.67	0.67

Mix	Ember2 Bit Error Rate							
	LA	LA+FR	FLA	FLA+FR	LA+AC	LA+AC+FR	FLA+AC	FLA+AC+FR
100/0	0.37	0.34	0.35	0.34	0.26	**0.25**	**0.25**	**0.25**
50/50	0.49	0.51	0.52	0.54	0.47	0.47	0.35	**0.32**
10/90	0.64	0.66	0.81	0.81	**0.55**	0.63	0.77	0.63
0/100	0.72	0.72	0.8	0.8	**0.67**	**0.67**	**0.67**	**0.67**