2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

A GPT-2 Tabular Transformer Model for Automated DRAM Verification

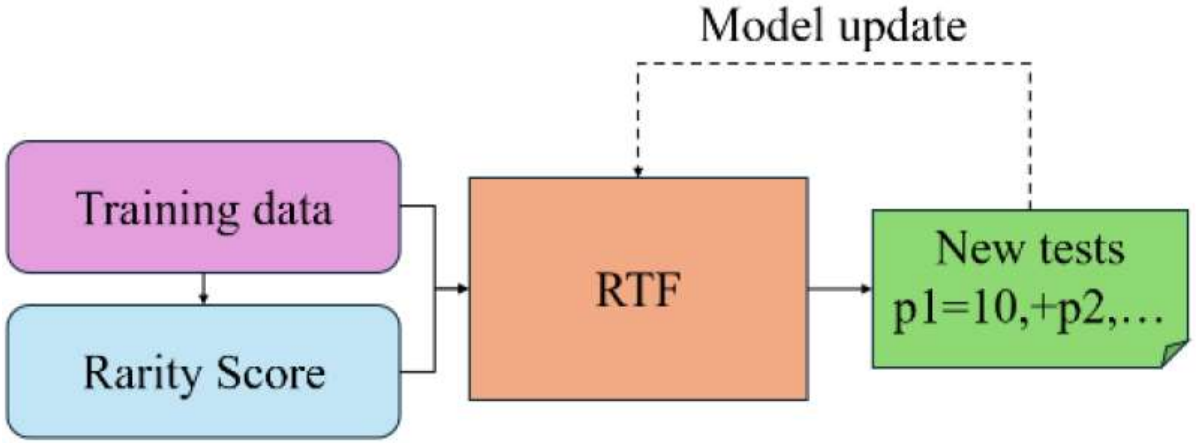
Lorenzo Ferretti, Chinmaya Behera, Shailesh Sharma, Nihar Athreyas, Vikram Narayan, Samir Mittal

Micron Technology

Motivation: DRAM Verification Challenges

- Verifying complex hardware designs such as DRAM is highly time consuming and resource intensive.
- Bus Functional Models simulate DUT behavior and allow engineers to evaluate how input sequences affect the design.
- Plusargs control the generation of these input sequences in a UVM testbench—selecting test types, enabling/disabling features, setting random seeds, and generally shaping DUT behavior.
- By exploring the plusargs space, DV engineers generate diverse test scenarios to maximize BFM functional coverage and thoroughly exercise the DUT.
- **Challenge:** The plusargs space is extremely large and cannot be exhaustively explored. Engineers rely on manually crafted or heuristic-based combinations to produce meaningful tests.
- Real DRAM designs require exploring ~10^5–10^6 coverbins, and plusargs create a combinatorial explosion of possible configurations, while simulation budgets limit how much can actually be run.

Our Solution: Automated Test Generation Leveraging Transformers Models



- GPT-2 Tabular Transformer (RTF) learns plusarg distributions.
- Conditional sampling targets rare functional bins.

Problem Formulation

Goal: maximize functional coverage within a fixed BFM budget.

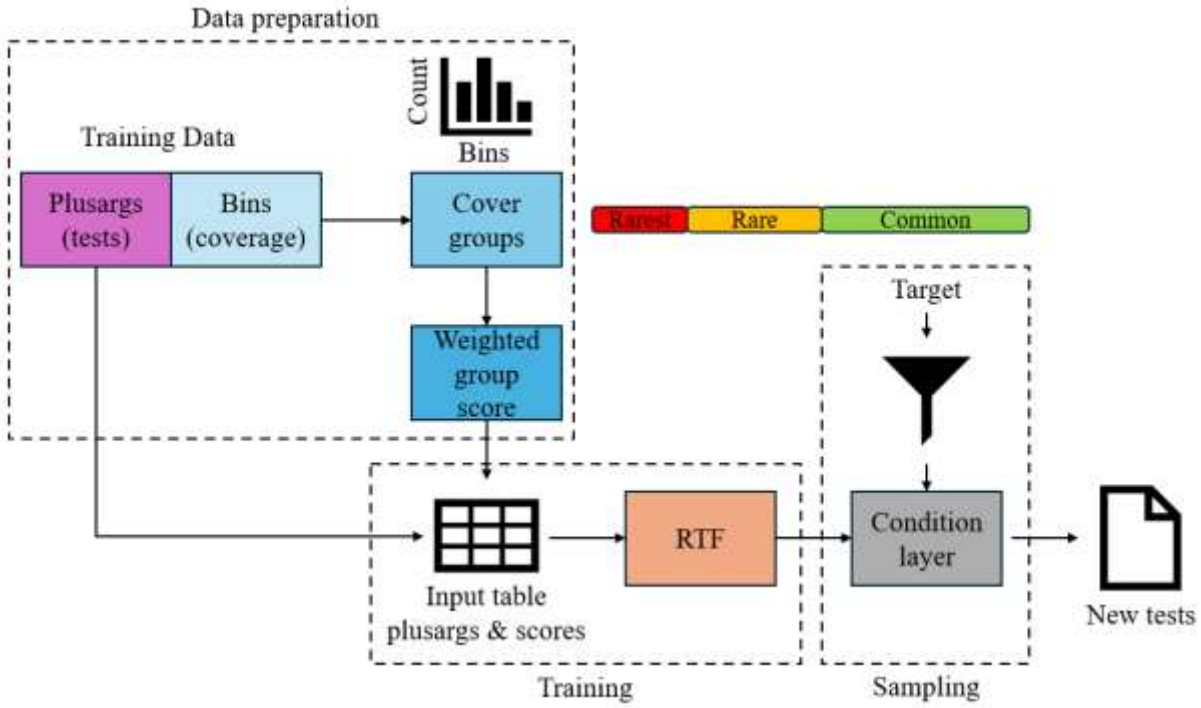
Inputs: plusargs, coverage vectors, contributing tests.

Output: new legal plusarg combinations likely to improve coverage.

RTF-Based Flow Overview

The proposed methodology is divided into four main phases:

1) **Data Preparation**: Use past simulation data to extract plusargs and coverbin hit frequencies. Compute a score from this data to define rarity groups and identify target bins.
2) **Model Training:** Convert tests and scores into a tabular format and use them to train the RTF model.
3) **Conditional Sampling:** Generate synthetic tests from the trained model, using a conditional layer to guide sampling toward distributions aligned with the target bins.
4) **Simulation**: Run regressions on each new batch of generated tests and iterate until the simulation-budget limit is reached.



Data Preparation

- Contributing tests are derived from simulation results obtained over the past weeks. Each prior week's simulation data is merged, and the contributing tests from the past weeks are extracted consecutively.
- Each test is associated with a set of plusargs that uniquely define the test. Imputation, smoothing, and data cleaning methods are applied to ensure that the input data is accurate and reliable.
- For each test, coverage information at a coverbin level is obtained from its simulation. For each coverbin, we store the coverage information with respect to each test across repetitions. This provides us with the frequency of coverage for a given coverbin *b* across a set of tests *t*. The coverage data is stored as a vector.
- For each test, the coverage vector indicates the bins hit during the test simulation. From the hit frequency, we derive different groups of bins:
  - Rarest: bins hit by 150 tests or fewer, including bins never hit (n <= 150)
  - Rare: hit by more than 150 tests but fewer than 400 tests (150 < n <= 400)
  - Common: hit by more than 400 tests (400 < n <= 1000).
- Based on this grouping, we can determine if each coverbin belongs to the rarest, rare, or common group, and calculate the rarity score as follows:

$$\text{RarityScore} = \frac{\#\text{ of bins hit by the test in group } i}{\text{Total number of bins}}$$

- where *i* in *G*, *i* identifies the group, and *G* is the set of all groups *G* = {Rarest, Rare, Common}. The rarity score within the plusargs are then used to define the input data for the RTF Training phase.

RTF Model Architecture

We leverage a GPT-2 architecture, inspired by the work from

- GPT-2 autoregressive transformer for tabular data. The model aims at learning the distribution across column (features) and values.
- Tokenization of plusarg-name + value is used to train the model of each input.
- Regularization prevents overfitting.

Tabular Representation

Input data plusargs and the associated values are represented as a Base matrix $M_{n \times p}$, where n is the number of test used to train the model and p the number of plusargs.

This is expanded with the rarity information, and the rarity score associated to each test, expanding the original M matrix to $M'_{n \times (p+3)}$.

Conditional Sampling

- We sample plusarg combinations based on input constraints and a Rarity Score, exploring the valid plusarg space according to the learned distribution.
- To improve coverage, we skew the sampling toward regions of the coverage landscape that are rarely exercised.
- A conditional sampling layer enriches training data by encoding plusargs, rarity scores, and two additional fields:
  - Relation (logical operators such as =, <, >),
  - Threshold (a value used with the relation).
- These fields capture how relational operators and thresholds correlate with plusarg combinations and rarity scores.
- During training, the model learns these distributions; during sampling, we provide the relation and threshold so the model generates plusarg combinations expected to exceed a desired rarity score.
- Users can select different relational conditions to target specific rarity groups.
- We use a hierarchical rarity definition (rarest, rare, common). The model is conditioned to generate tests more likely to fall into the rarest group, increasing the likelihood of hitting bins that are seldom exercised.

Simulation Loop

- Once the new batch of tests is sampled, the tests are simulated, and the coverage information is extracted.
- After each batch of simulation is completed, the coverage outcome of the simulated data is added to the training corpus.
- The model is then re-trained using the same process. This cycle is repeated until a user-defined number of simulations is reached.

Experimental Setup

Evaluation Setup

- Tested our methodology on a legacy industrial DRAM design (~497K functional bins).
- After excluding unreachable/infeasible scenarios, remaining bins defined the coverage target for the simulation budget.

Industry AI Flow (Baseline)

- Weekly flow using prior-week training data.
- Produces 6 runs/week, each with 1,000 simulations.
- Best run is selected for hardware simulation and reused as training for the next week.
- This baseline matches the flow published in [1], which we replicated for fair comparison.

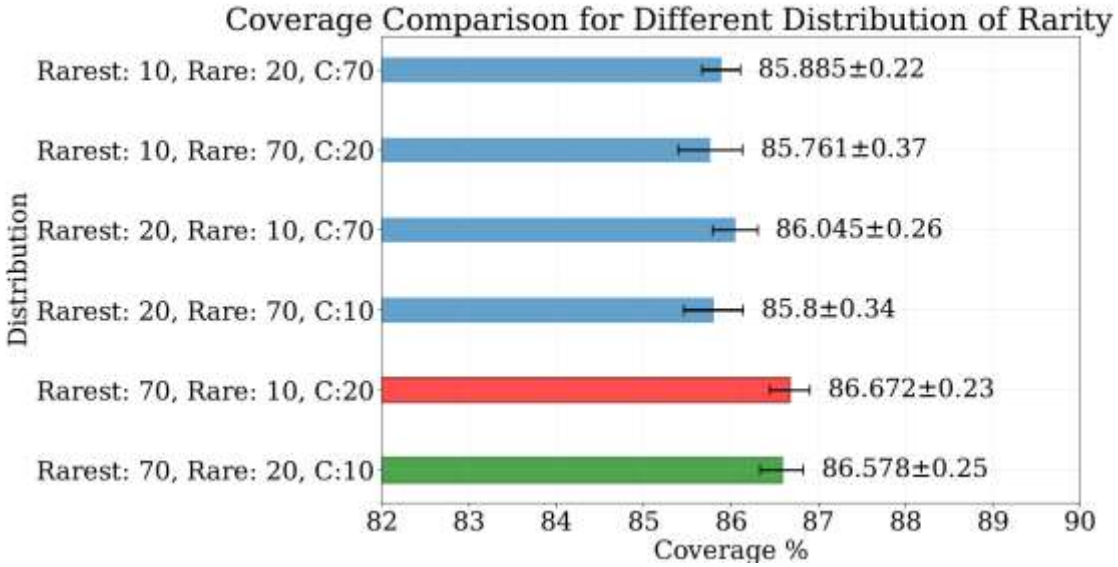
Our Methodology

- Trained the RTF model on the same prior-week coverage data as the industry tool.
- Generated a separate simulation batch using our flow for direct, apples-to-apples comparison.

[1] G. Pallela, P. Shi, S. Deshmukh, V. Venkatesh, B. Hughes, R. Suvarna,and S. Srinivasan, “Stimulus diversification and coverage closure of 3dnand flash with artificial intelligence,” in DVCon, 2025.

Experimental Setup

- Initially, contributing tests are collected from historical data.
- Rarity scores for each of the groups (Rarest, Rare, and Common) are calculated.
- These scores, along with plusargs, are used to train the RTF model.
- In the current implementation, 70% of the samples are expected to belong to the Rarest group, 10% to the Rare group, and 20% to the Common one. These numbers were chosen based on empirical observation.
- Lastly, the sampled tests produced from RTF are adjusted to factor out randomness to avoid getting stuck in local optima. Currently, 10% of tests are randomly generated.



Empirical exploration of sampling percentage of each rarity group. The red and green bars are the best performing ones, averaged across 5 repetitions. Sampling distributions are indicated as Rarest, Rare, and Common (C).

RTF Statistical Performance

Model Learning Assessment

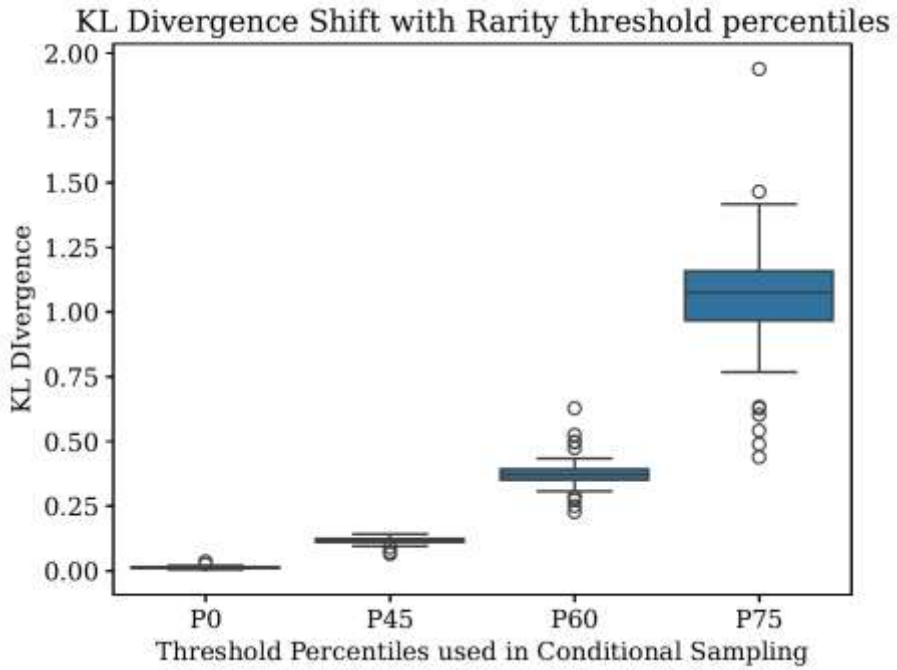
- Measured how well the RTF model learns plusargs and legal value distributions using marginal distribution similarity for each column.
- Achieved 96.15% marginal similarity and 97.11% column-correlation similarity, showing strong alignment with real data.

KL Divergence Analysis

- Evaluated KL divergence between training data (P0) and samples conditioned at P45, P60, and P75 thresholds.
- Higher thresholds show increased KL divergence due to smaller sample regions and distribution skew.
- P0 shows low KL divergence, confirming accurate reproduction of training data.
- Increasing thresholds shifts sampling toward targeted high-rarity regions in plusargs space.

Key Insight

- Model effectively learns real distributions and selectively samples harder-to-reach regions—supporting improved coverage of rarely hit bins.



RTF Statistical Performance

Conditioning Layer Evaluation

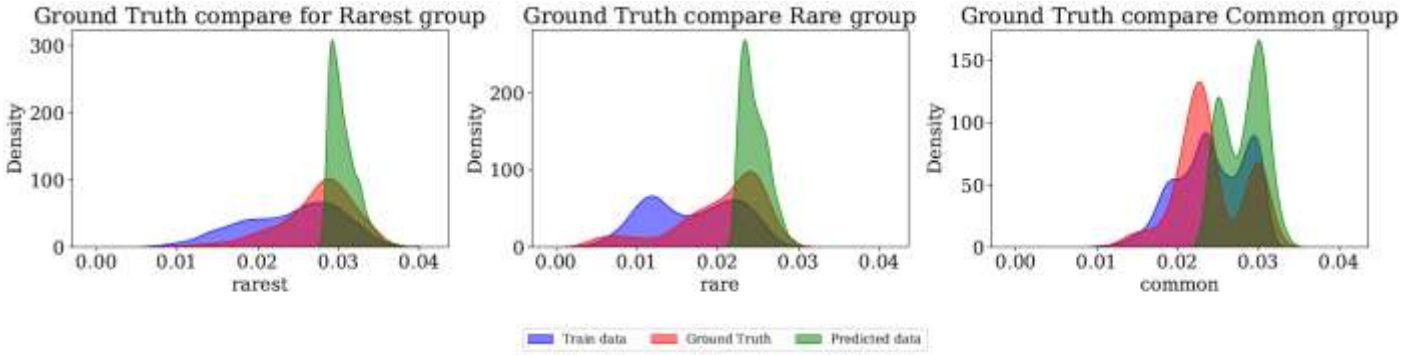
- Assessed how well the conditioning layer skews test generation to match desired rarity-score groups (rarest, rare, common).
- Compared training distribution (blue), model-predicted distribution (green), and ground-truth simulation results (red) for each rarity group.
- Results show consistent alignment: predicted distributions closely mirror ground truth, confirming effective conditional sampling.

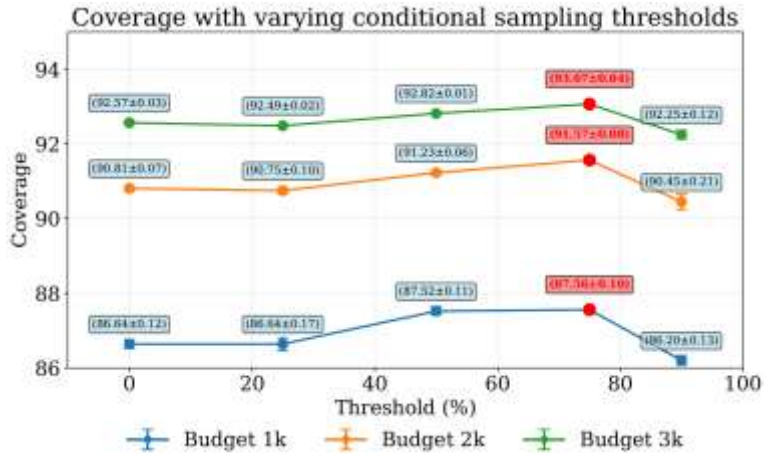
Threshold Selection

- Evaluated multiple threshold conditions and simulated resulting tests.
- 75% threshold delivered the best coverage performance across simulation budgets.

Key Insight

- The conditioning layer accurately learns and reproduces nuanced rarity-based distributions, enabling generation of targeted tests that better reach hard-to-hit bins.

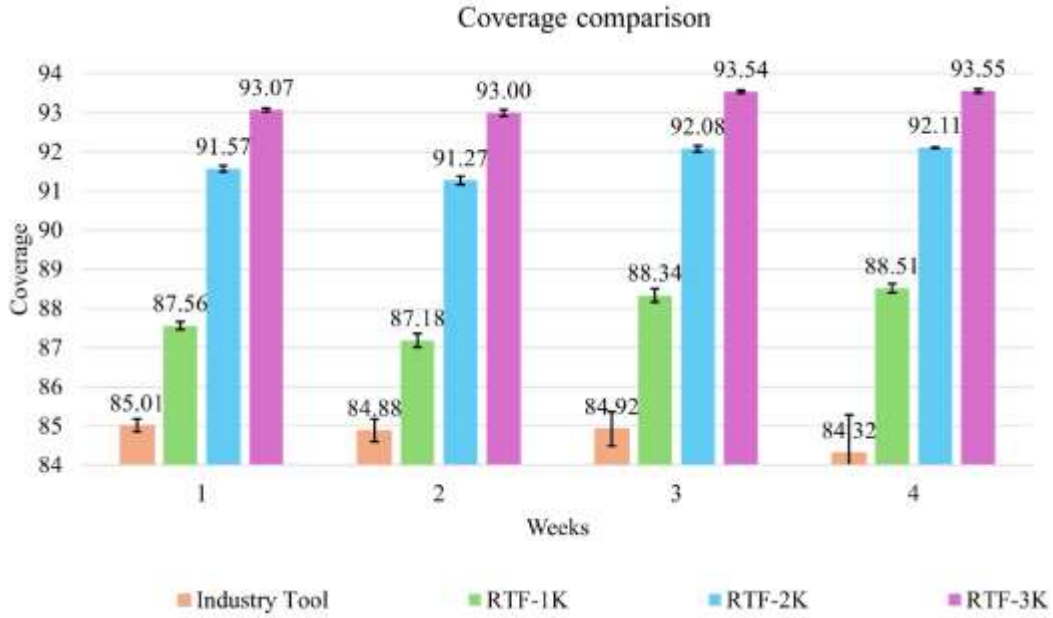




Coverage Improvement

Weekly Comparison: Industry AI Tool vs. RTF

- Evaluated both flows across four dynamic weeks, where design and testbench changes naturally occurred.
- Week 1: Both flows start with the same information.
- Weeks 2–4: Industry AI tool builds on tests it previously discovered whereas RTF flow builds on tests generated by its own exploration.



Results

RTF consistently outperforms the industry AI tool across all weeks.

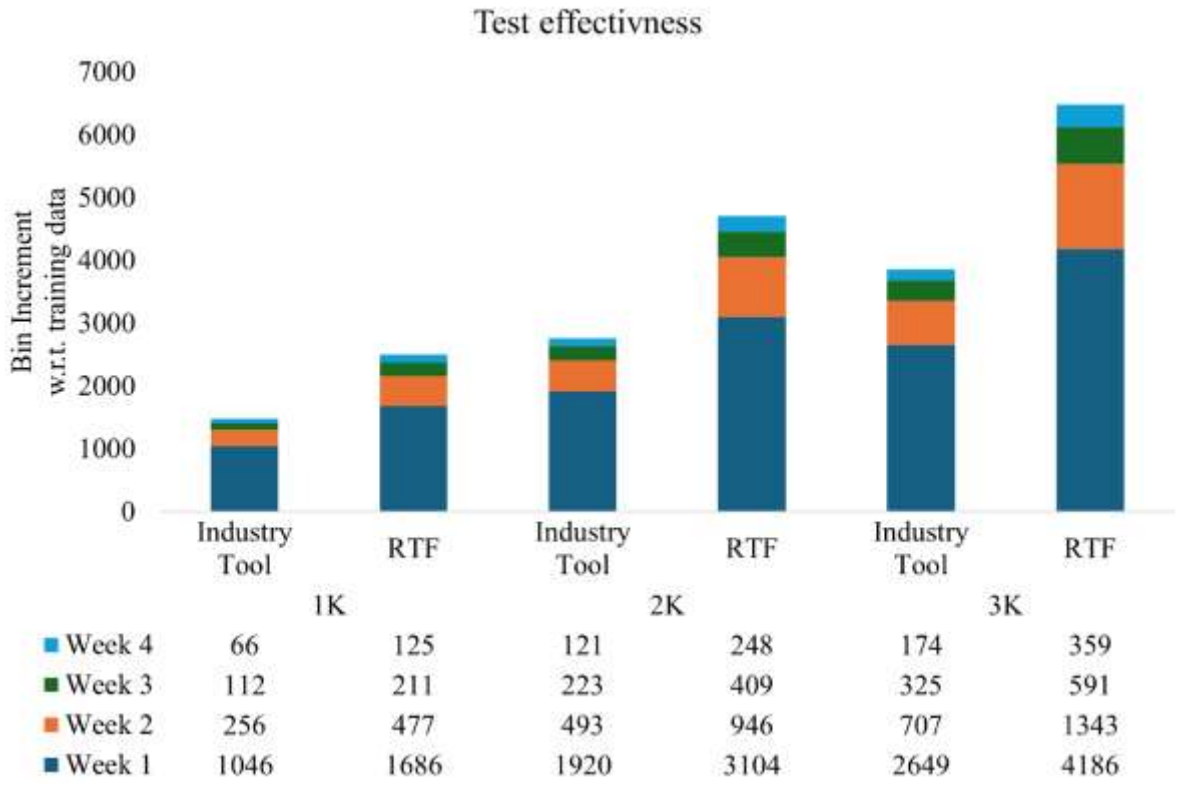
After four weeks, RTF achieves higher average coverage gains:

RTF-1K: +4.2% RTF-2K: +7.8% RTF-3K: +9.2%

RTF shows smaller standard deviation, indicating more reliable and stable performance than the industry AI tool.

Test Effectiveness

- In addition to end-of-regression coverage, we measured each flow's ability to discover new bins relative to the initial training data.
- This test-effectiveness metric evaluates how well each flow explores new regions of the search space rather than re-sampling already learned areas.
- Week 1: Both flows start from the same training data.
- From Week 2 onward: Each flow relies on its own prior-week discoveries to guide the next week's exploration.



Model Ensemble

RTF + Industry AI Tool (Ensemble Evaluation)

- Assessed the combined benefit of merging RTF and industry AI tool results under the same simulation budget.
- Multiple industry AI tool runs were merged to ensure a fair comparison with RTF-1K, RTF-2K, and RTF-3K.

Discovery of Unique Scenarios

RTF discovers significantly more unique bins than the industry tool:

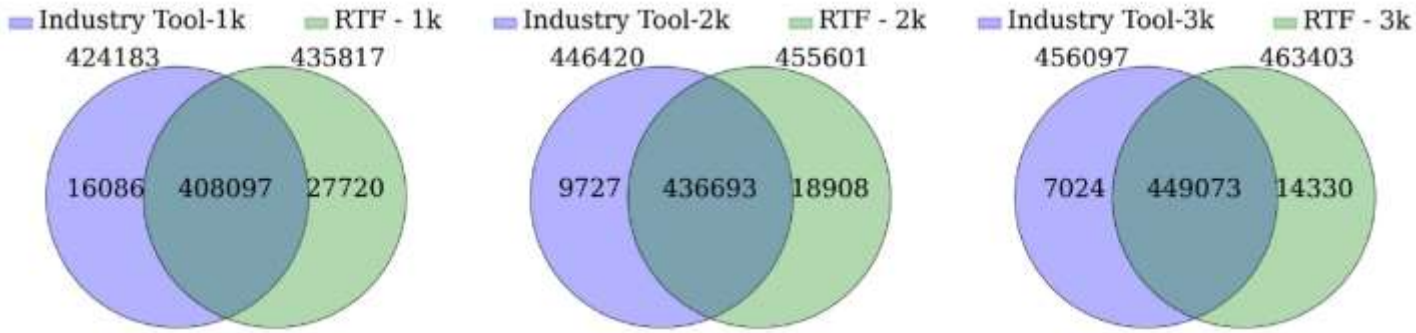
- RTF-1K: 1.7× more unique bins
- RTF-2K: 1.9× more
- RTF-3K: 2.0× more
- Industry AI tool also contributes unique bins, indicating complementary exploration patterns.

Ensemble Coverage Benefit

Merging both flows yields higher total coverage:

90.92%, 93.62%, 94.65% for 1K, 2K, and 3K budgets.

The two flows explore different regions of the search space; combining them leverages their complementary strengths.



Conclusion

- The proposed methodology presents a comprehensive approach to accelerate the verification of DRAM hardware designs by generating new tests using a conditioned transformer model architecture.
- By leveraging the strengths of transformer models and incorporating a novel rarity score feature, this technique provides a targeted and efficient way to ensure thorough testing and validation of complex hardware systems, achieving up to 7.89% coverage improvement with respect to the currently employed design verification flow.