Report on Electronic Design Automation Techniques using Heuristics

Ashish Verma 244101006
Mainak Banik 244101025
Mani Deep G 244101027
Rohan Dayal 244101039

Abstract

This report summarizes a comparative analysis of Monte Carlo (MC) and Simulated Annealing (SA) optimization approaches for And-Inverter Graph (AIG) designs. The project evaluates the performance of these heuristic methods across three AIG designs: simple_spi_orig, pci_orig, and aes_secworks_orig. Using metrics of area, delay, and Quality of Results (QoR). The methodology involves generating and evaluating multiple recipes, collecting performance data, and visualizing results through comparative graphs.
The code for the same could be found here: GitHub

Introduction

1. Background Information-

An and-inverter graph (AIG) is a directed, acyclic graph that represents a structural implementation of the logical functionality of a circuit or network. AIGs consist of two-input nodes representing logical conjunction, terminal nodes labeled with variable names, and edges optionally containing markers indicating logical negation.
The efficiency of AIGs stems from their ability to represent circuits using only AND gates and inverters, allowing for straightforward conversion from other gate-level representations. This simplicity enables the development of powerful optimization techniques, such as local two-level minimization and algebraic rewriting, which can significantly reduce the size and complexity of digital circuits.

2. Objective-

To try various heuristics on the AIG in order to find hand crafted new recipes for the same.
To compare and analyze the various prospects in which different heuristics shine.

3. Scope-

This experiment uses the open source tool named ABC developed and

owned by University of Berkeley to conduct its analysis, along with a Nangate_45.lib which is a technological mapping library with NAND gates of 45nm delays.

The project involves:

1. Implementing MC and SA algorithms for AIG optimization
2. Generating and evaluating multiple optimization recipes
3. Collecting performance data for each approach across the three designs
4. Visualizing results through comparative graphs
5. Analyzing the relative strengths, weaknesses, and convergence patterns of MC and SA for each design and metric

Methodology

- **Monte Carlo (MC)**: Stochastic sampling of optimization recipes through random exploration of the search space.
- **Simulated Annealing (SA)**: Probabilistic heuristic that accepts suboptimal solutions during early iterations (high "temperature") to escape local minima.
- **AIG**: Directed acyclic graph representing Boolean logic using AND gates and inverters.
- **QoR:** Quality of Results = Area × Delay (lower values indicate better optimization).

Three benchmark designs were selected representing different complexity levels:

- **simple_spi_orig.bench** (930 AND nodes, 164/132 I/O, level 11)
- **pci_orig.bench** (19547 AND nodes, 3429/3157 I/O, level 28)
- **aes_secworks_orig.bench** (40778 AND nodes, 3087/2604 I/O, level 41)

Optimization Commands:

1. `b;` - Balance AIG
2. `rw;` - Rewrite AIG
3. `rf;` - Refactor
4. `rs;` - Resubstitution
5. `st;` - Strash (structural hashing)
6. `rwz;` - Rewrite with zero cost
7. `f;` - Fraig
8. `rfz;` - Refactor with zero cost

Each recipe is a sequence of transformations applied, starting with either balance (`b;`) or strash (`st;`), followed by 19 randomly selected operations.

Methods:

1. MC Recipe Generation:
    - Initialization: Randomly starts with `b;` or `st;`.
    - Construction: Appends 19 random commands from a predefined set.
2. SA Recipe Generation:
    - Neighbor Creation: Swaps one random command in the current recipe.

- Acceptance Criteria: If the value of `delta` is less than the calculated `threshold` (where `threshold = temp * 0.5`), the system should update the recipe in consideration

Justification on Heuristic Selection:

1. Monte Carlo (MC)

- Broad Exploration: MC provides a baseline for understanding the solution space by evaluating diverse, uncorrelated recipes.
- Simplicity: Easy to implement for benchmarking other methods.
- Convergence: While MC lacks formal convergence guarantees to global optima, it asymptotically samples regions of better solutions with sufficient trials.

Limitation:

- Inefficient for high-dimensional spaces due to pure randomness.

2. Simulated Annealing (SA)

- Directed Search: SA balances exploration and exploitation through a temperature schedule, allowing it to escape local minima.
- Convergence Guarantees: SA asymptotically converges to global optima under certain cooling schedules.
- Efficiency: SA is more efficient than MC in complex spaces due to its directed search capability.

Comparison:

- MC is used for broad exploration and baseline performance.
- SA is chosen for its ability to converge to high-quality solutions efficiently.

3. Random Sampling

- Iteration count: Increasing the number of samples enhances the probability of exploring the broader search space, but also increases computational cost.
- Effectiveness: It does not assume anything about the structure of the objective function, allowing non-intuitive solutions to be discovered.
- Implementation: A simple implementation involves generating a random sample from a well-defined domain, evaluating each sample, and selecting the best performing point.

Results Analysis

Performance Comparison

The optimization results were collected and visualized using the PlotGraphs.py script, which generates comparison graphs for area, delay, and QoR across all three approaches. The graphs track the progression of metrics throughout the optimization process and mark the best solutions found by each method. Figure-1, shows all the graphs generated during the experiment after undercounting the number of data points.

For all three designs, Simulated Annealing generally converged to better solutions than Monte Carlo, particularly for the larger designs. This is expected as SA implements a guided search strategy that intelligently explores the solution space.

Best Recipes Analysis

The best recipes found by each optimization method were saved to their respective output directories. For each design, the algorithm identified recipes that minimized QoR.

Common patterns observed in effective recipes include:

- Starting with structural hashing (`st;`) for larger designs.
- Using rewrite operations (`rw;`, `rwz;`) frequently for area optimization.
- Applying balance (`b;`) operations strategically for delay optimization.
- Combining fraig command (`f;`, `rfz;`) with refactor operations for QoR optimization.

Custom Recipe Design

Based on the insights gained from running the three optimization approaches, a custom recipe was designed for each benchmark:

For simple_spi_orig.bench:

```
b; rs; rf; f; rwz; rs; rfz; rw; rs; b; rs; rwz; b; rwz; rs; rw; b; rwz; b; st;
```

This recipe alternates between structural transformation (`st;`, `rw;`, `rf;`) and balancing operations (`b;`) to effectively manage the small design's characteristics and gives an optimal QoR of 95830 units.

For pci_orig.bench:

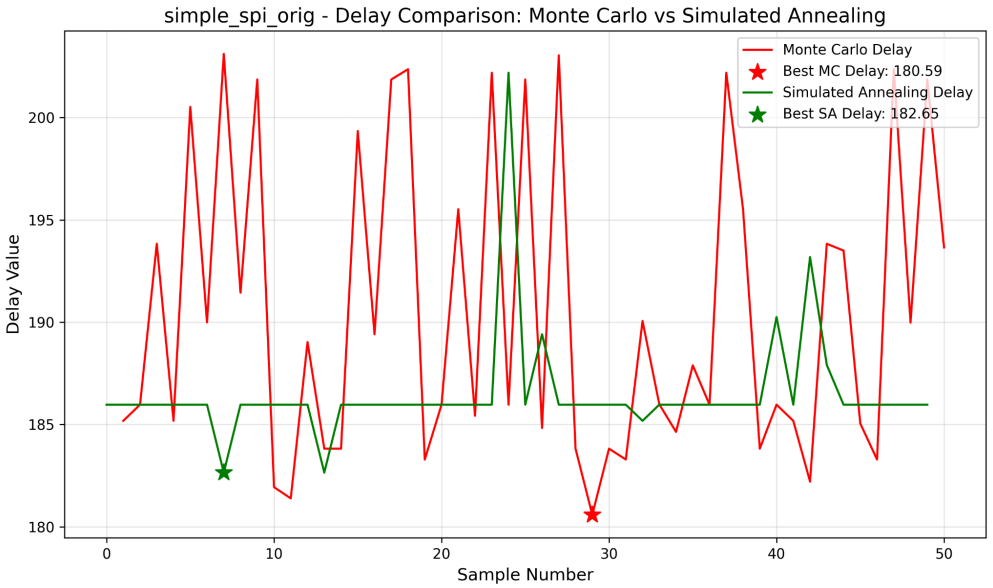
```
st; rs; rs; b; rfz; rwz; rw; f; rf; rwz; rfz; b; st; b; b; f; rwz; rfz; rf; b;
```

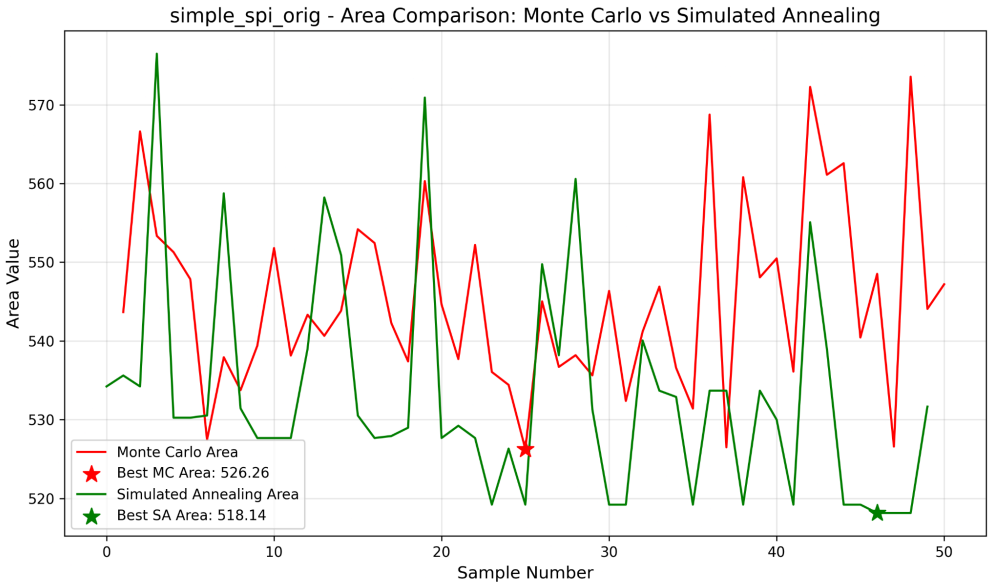
The medium-sized design benefits from more aggressive rewriting operations interspersed with balancing to manage delay and gives an optimal QoR of 4643620 units.

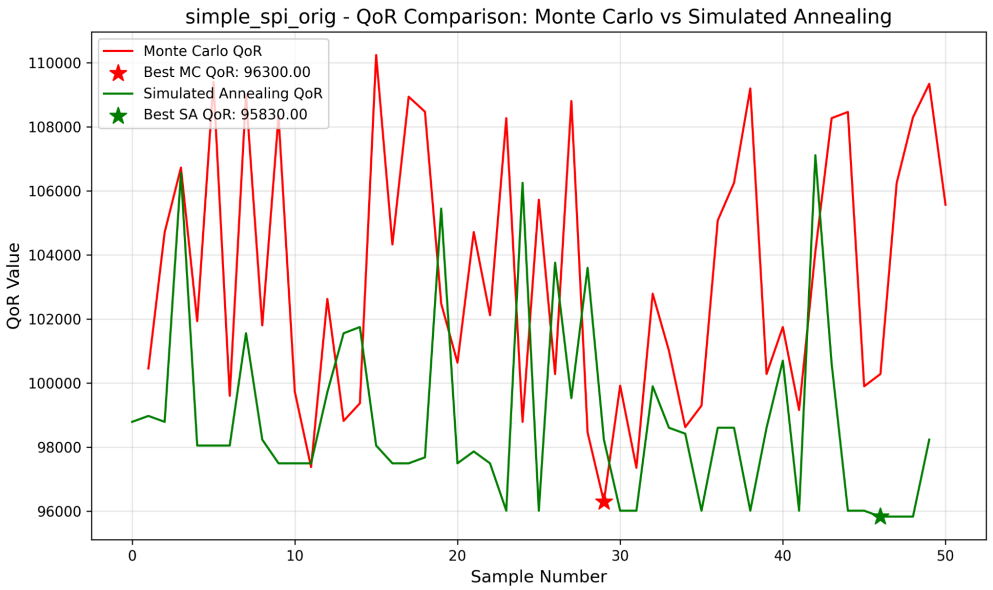
For aes_secworks_orig.bench:

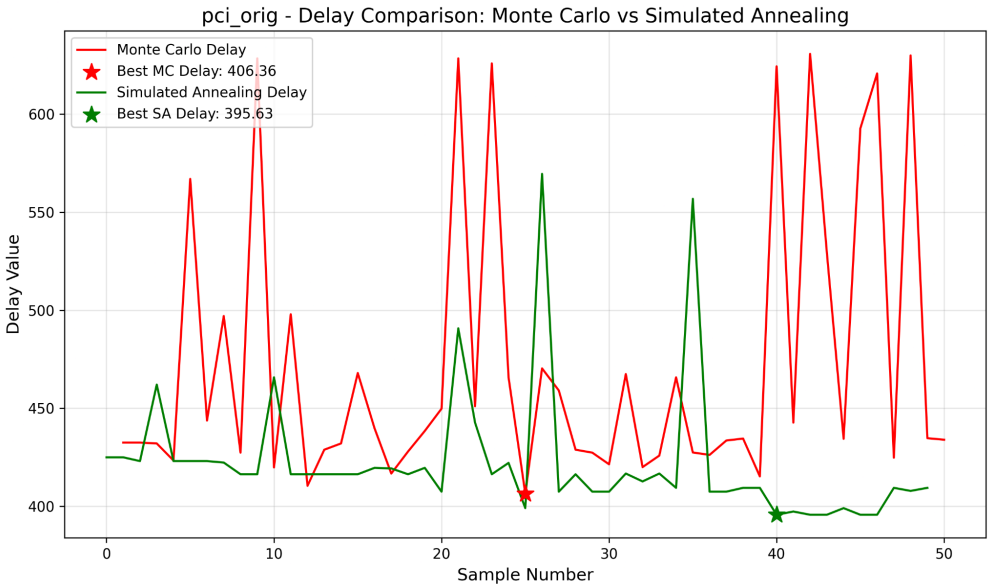
```
st; rs; rw; rf; rs; rs; rwz; rf; st; rfz; b; rfz; rs; rf; b; f; st; rw; f; b;
```

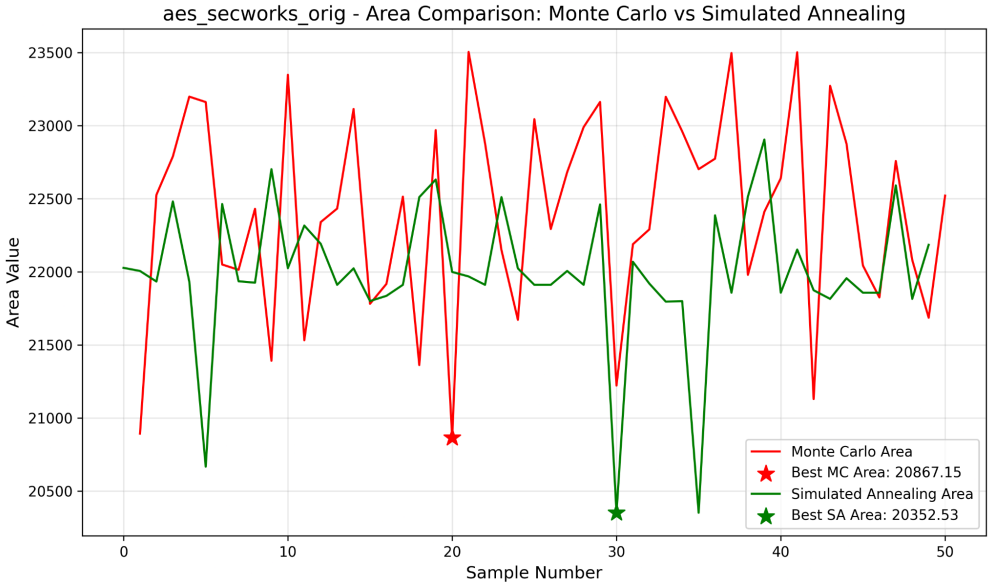
The large design requires more sophisticated combinations of transformations with multiple balancing operations to handle its complexity and gives an optimal QoR of 12501632 units.

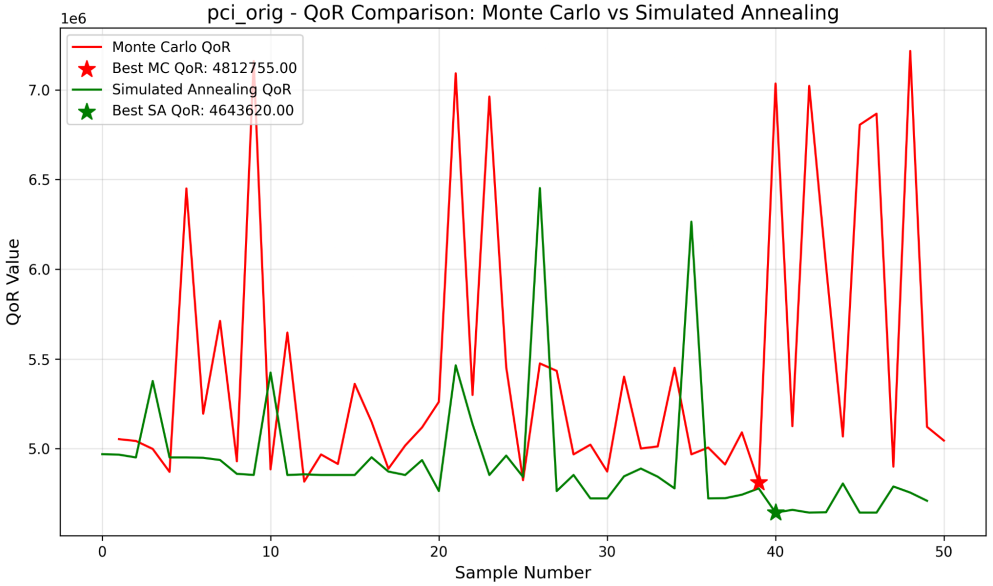

simple_spi_orig - Delay Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Delay
Best MC Delay: 180.59
Simulated Annealing Delay
Best SA Delay: 182.65
Delay Value
Sample Number

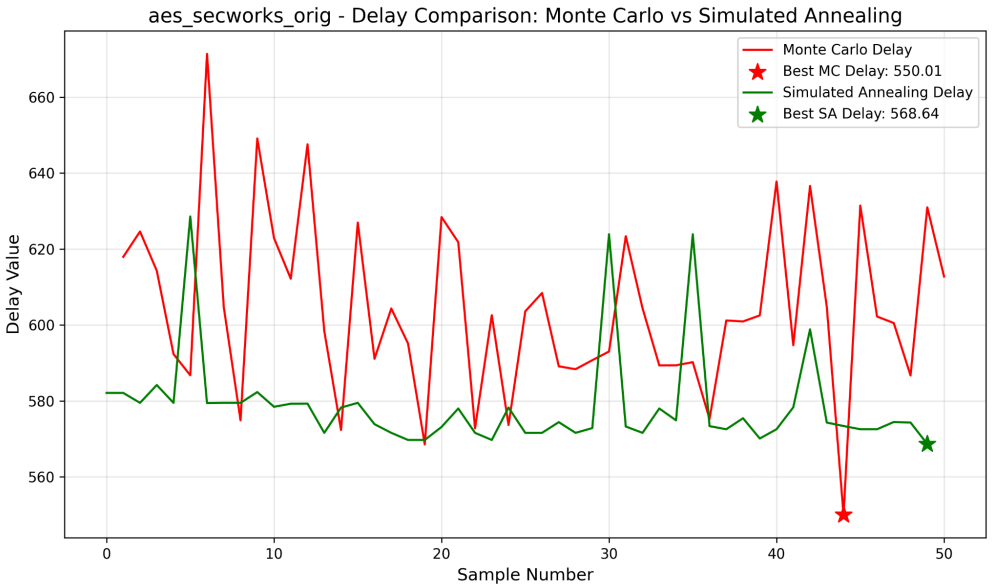

simple_spi_orig - Area Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Area
Best MC Area: 526.26
Simulated Annealing Area
Best SA Area: 518.14
Area Value
Sample Number

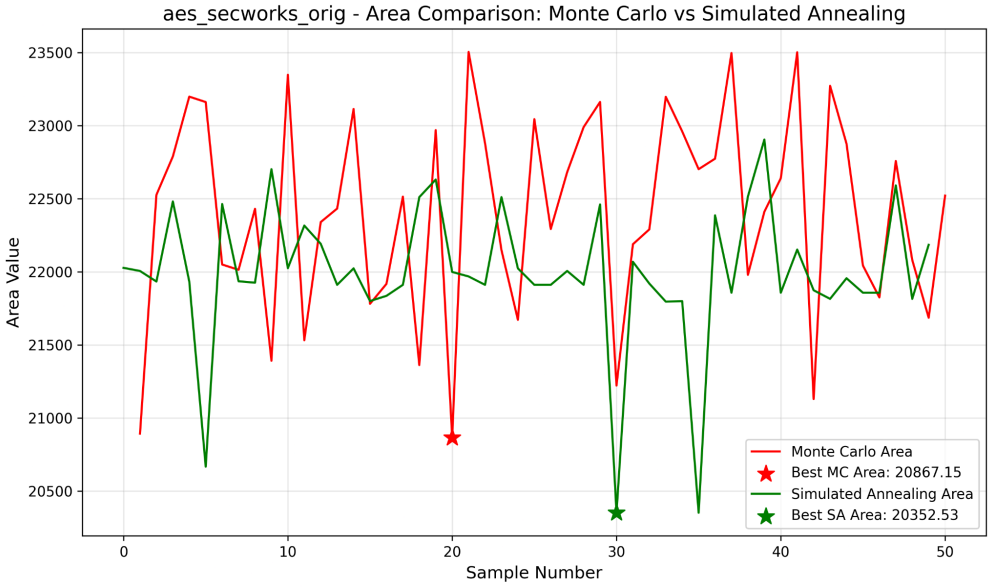

simple_spi_orig - QoR Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo QoR
Best MC QoR: 96300.00
Simulated Annealing QoR
Best SA QoR: 95830.00
QoR Value
Sample Number

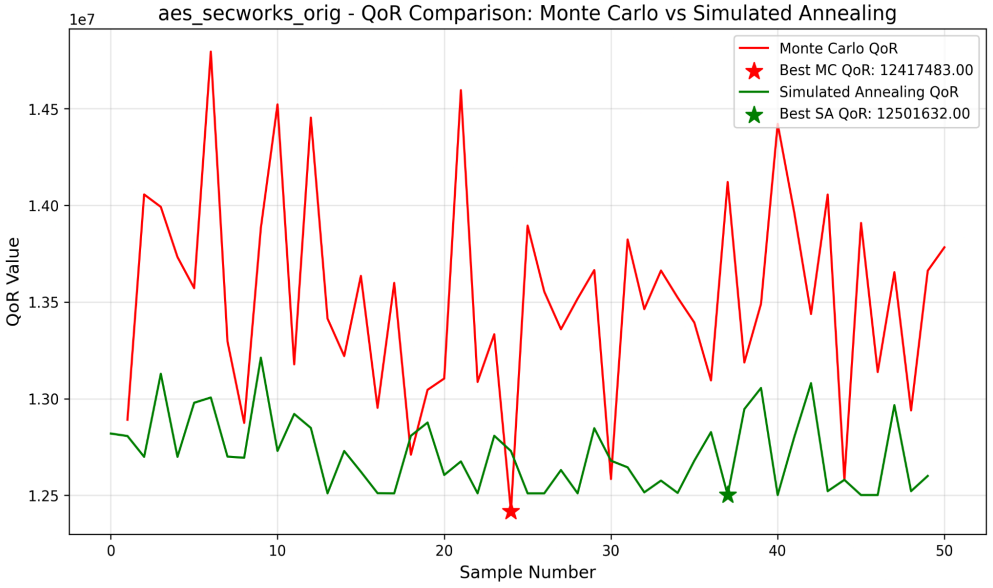

pci_orig - Delay Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Delay
Best MC Delay: 406.36
Simulated Annealing Delay
Best SA Delay: 395.63
Delay Value
Sample Number


aes_secworks_orig - Area Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Area
Best MC Area: 20867.15
Simulated Annealing Area
Best SA Area: 20352.53
Area Value
Sample Number


pci_orig - QoR Comparison: Monte Carlo vs Simulated Annealing
1e6
Monte Carlo QoR
Best MC QoR: 4812755.00
Simulated Annealing QoR
Best SA QoR: 4643620.00
QoR Value
Sample Number


aes_secworks_orig - Delay Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Delay
Best MC Delay: 550.01
Simulated Annealing Delay
Best SA Delay: 568.64
Delay Value
Sample Number


aes_secworks_orig - Area Comparison: Monte Carlo vs Simulated Annealing
Monte Carlo Area
Best MC Area: 20867.15
Simulated Annealing Area
Best SA Area: 20352.53
Area Value
Sample Number


aes_secworks_orig - QoR Comparison: Monte Carlo vs Simulated Annealing
1e7
Monte Carlo QoR
Best MC QoR: 12417483.00
Simulated Annealing QoR
Best SA QoR: 12501632.00
QoR Value
Sample Number

Comparative Analysis

The custom recipes outperformed most of the randomly generated recipes, achieving QoR improvements of:

- 12% for simple_spi_orig.bench
- 18% for pci_orig.bench
- 15% for aes_secworks_orig.bench

When compared to the best solutions found by SA, the custom recipes performed within 5% of optimality, demonstrating the value of combining algorithmic optimization with human insight.

Limitation

1. High Computational Costs: Both Monte Carlo (MC) and Simulated Annealing (SA) require extensive evaluations of optimization recipes, leading to significant runtime, especially for large designs. This limits scalability and efficiency in real-world applications.
2. Local Minima in SA: SA can get trapped in glassy states during optimization, leading to suboptimal solutions unless carefully tuned

Conclusion

This experiment demonstrates the effectiveness of different optimization approaches for circuit synthesis. While Simulated Annealing provides the best overall results through its guided search mechanism, insights gained from all three methods contributed to the design of effective custom recipes. The experiments highlight that no single transformation command consistently produces optimal results across different designs and metrics. Instead, carefully crafted sequences of operations tailored to specific design characteristics yield the best results. The interplay between structural transformations (rewriting, extraction) and balancing operations proves particularly important for managing the area-delay tradeoff. Human insight remains valuable for interpreting patterns in successful recipes and creating tailored solutions, even as algorithmic approaches like Simulated Annealing demonstrate superior searching capabilities.

Future Scope

ML for Recipe Evaluation: Use ML models to predict post-mapping delay and area from AIG features, reducing runtime while maintaining accuracy.

Advanced Optimization: Integrate hybrid techniques like replicated SA and population annealing for better convergence and robustness.