CMPEN 431 Final Project Report

Design-Space Sequencing Heuristics for EDP and ED^2P

Avie Vasantlal

Spring 2025

Abstract

In this project, we explored different processor designs using a tool called SimpleScalar to see which setups work best under different metrics. We looked at four different goals throughout this project which were: EDP (Energy x Delay), ED^2P (Energy x $Delay^2$), EDAP (Energy x Delay x Area), and ED^2AP (Energy x $Delay^2$ x Area). I created rules (other known as heuristics), in order to help the computer pick better, calculated decisions instead of just choosing randomly. For EDP, I started with a simple, low-energy design and made minor improvements. For ED^2P, I started with a powerful setup and began removing components if they didn't have an impact on the delay. After testing 1000 designs for every metric, I found that the designs varied depending on the goal. Therefore, from this project we can assume that optimizing a processor depends on which improvement you want to target, whether that be saving energy, speed, or using less space.

1: Introduction

Modern processor design involves using a complex space of architectural choices. In this project, I used a eighteen-dimensional space consisting of features like issue width, cache sizes and associativity, branch predictors, buffer depths, and much more. Since each configuration consisted of multiple settings, we used heuristics to make educated choices on the parameters. This report explains how the heuristics were used to find good processor setups for four different configurations.

2. Methodology

2.1 Configuration Space

Each processor configuration is represented by a vector of 18 integers, with each index corresponding to one setting for one dimension. These dimensions include core execution parameters like fetch width, and scheduling style, memory hierarchy properties such as cache sizes, associativity, and block sizes, and microarchitectural components like register file and memory port sizing. All configurations were required to follow a set of rules, like ensuring that the L2 cache is at least as large as the combined L1 instruction and data caches, and that cache latency values match prescribed values based on size and associativity. The validity of these configuration spaces was done using the *validateConfiguration()* function.

2.2 Baseline Configuration

The baseline configuration is defined as:

[0,0,0,0,0,0,5,0,5,0,2,2,2,3,0,0,3,0]

Which corresponds to:

- Width: 1
- Fetch Speed: 1
- Scheduling: In-order
- RUU: 16 entries
- LSQ: 4 entries

- MemPorts: 1
- DLI/ILI/UL2 Cache: mid-sized, 2-way
- Latencies matched via lookup tables
- Perfect Branch predictor

2.3 Heuristic Proposal Function

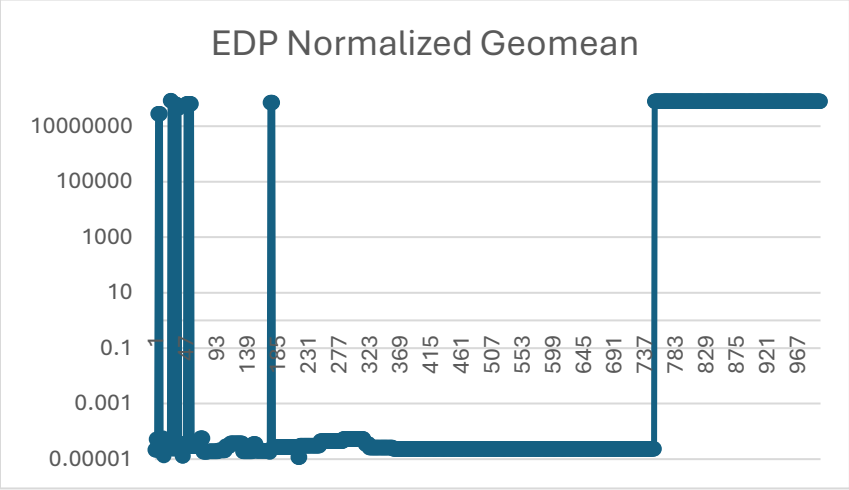
The *YourProposalFunction* generates settings that favor each metric:

- For EDP, I fixed parameters that increased energy use when scaled up. This resulted in minimal issue width, fast fetch, in-order scheduling and perfect prediction
- For ED^2P, I fixed parameters in order to encourage an aggressive performance-oriented setup. This had a width of 8, out-of-order scheduling and large buffer sized for RUU and LSQ.
- For EDAP, I fixed parameters that retained low-width and in-order execution to minimize area & energy. This meant that we had to remove the perfect prediction due to its area cost.
- For ED^2AP, I used a wide pipeline and out-of-order scheduling to prevent $delay^2$ penalties in order to not use large cache structures. The Idea was to reduce area in order to retain performance wherever possible.

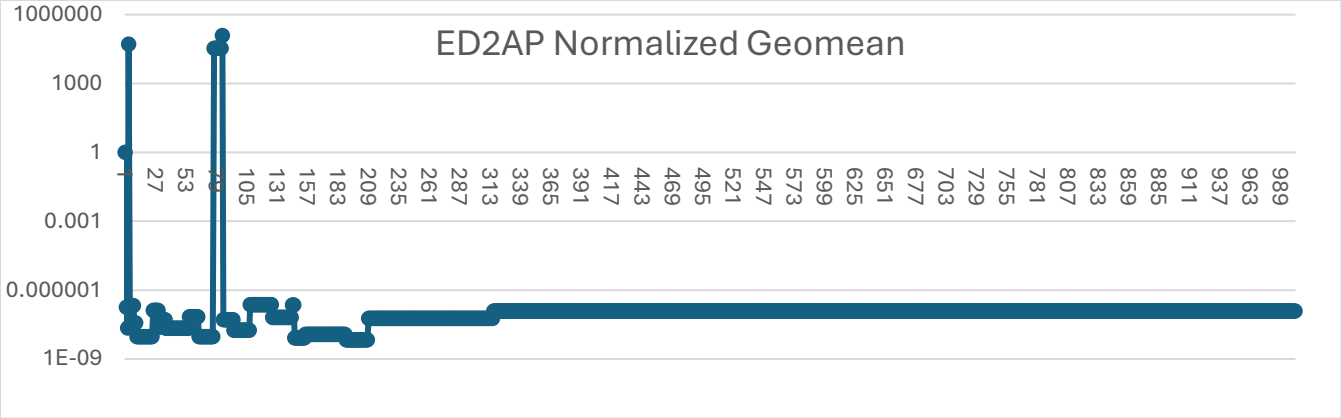
3. Results & Plots

3.1 Line Plots – Normalized Geomean vs. Design Count

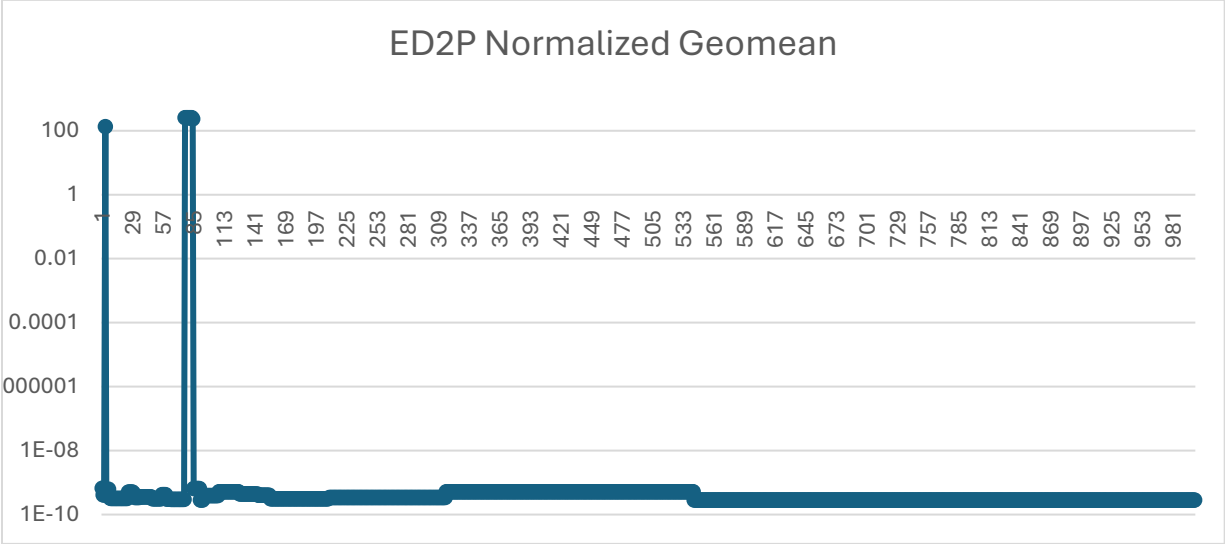
Each of the following line plots demonstrated how the normalized geomean of the optimization metric (EDP, ED^2P, EDAP, ED^2AP) changed as more design points were given and validated.



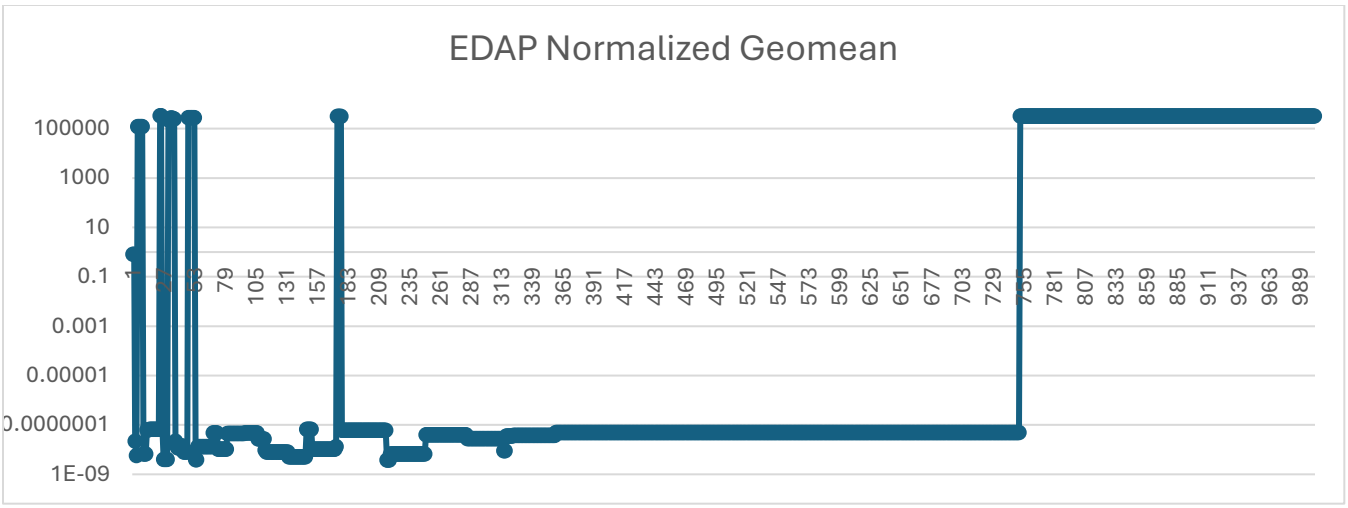
- EDP Line Plot: Initially, for the first 150 or so iterations, the heuristic consisted of some extremely high values implying that the parameters were either invalid, poorly optimized or had high penalties. After this stage of instability, the heuristic it hit a phase where it consistently outperformed the baseline due to the low-width, in-order execution, and compact caches which EDP favors. After the ~740th run the heuristic begins become instable which may have been due to invalid configurations which led to severe penalties.



- ED2AP Line Plot: For the initial 150 iterations it finding its exploration space, leading to occasional extreme spikes. However, afterwards it stays within the range of 10^{-6} to 10^{-8} and then flattens for the rest of the run implying that the heuristic locked into a configuration space that reliably produced highly efficient and compact architectures under the metric. This plot demonstrates a very successful heuristic. The beginning may have had some outliers and instability but the rest of the heuristic executed with precise effective efficiency. Since all values after ~200 fall below 10^{-6}, it is at least a million times better than the baseline. This was most likely achieved due to moderate cache sizes that met the constraints but avoided large area overhead.



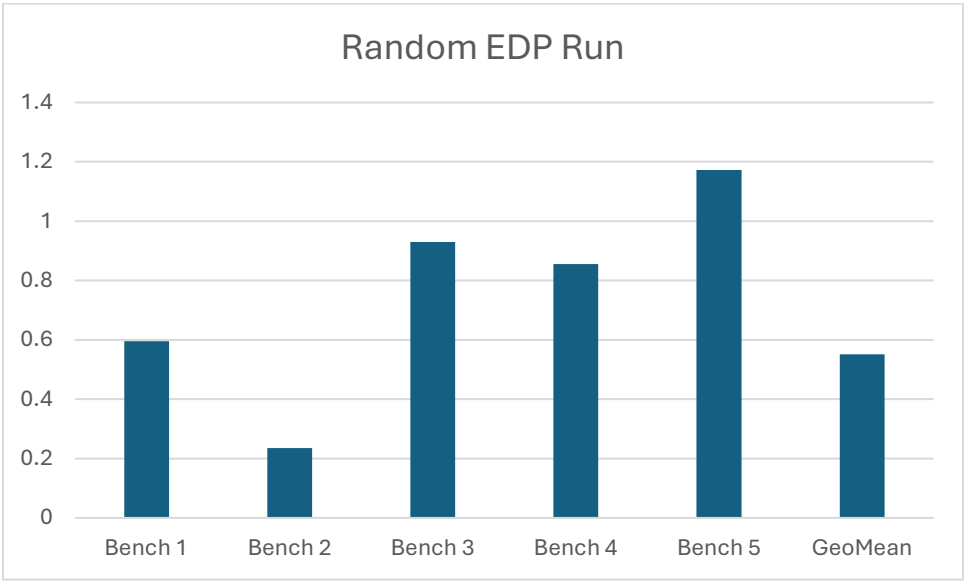
- ED2P Line Plot: This chart indicated the ED^{2}P heurtistic was very effective. After a few short spikes it found high-performing configurations that remained constant for the rest of the run. The squared delay term pushed the search more towards maximum throughput configurations and the lack of spikes later in the plot indicates that the strategy avoided inefficient designs. The values produced were between 10^{-9} and 10^{-10} implying that the heuristic performed significantly better than the baseline and was extremely efficient.



- EDAP Line Plot: This geomean plot shows that the heuristic eventually found a productive region but took a while to get there and did not stay there. The middle of the run had consistent results which was a significant success. However, the poor start and end demonstrate a higher variance than the other searches. The volatility suggests that EDAP requires more modification of the heuristic to maintain the behavior that was demonstrated during the 200-700 iteration interval.

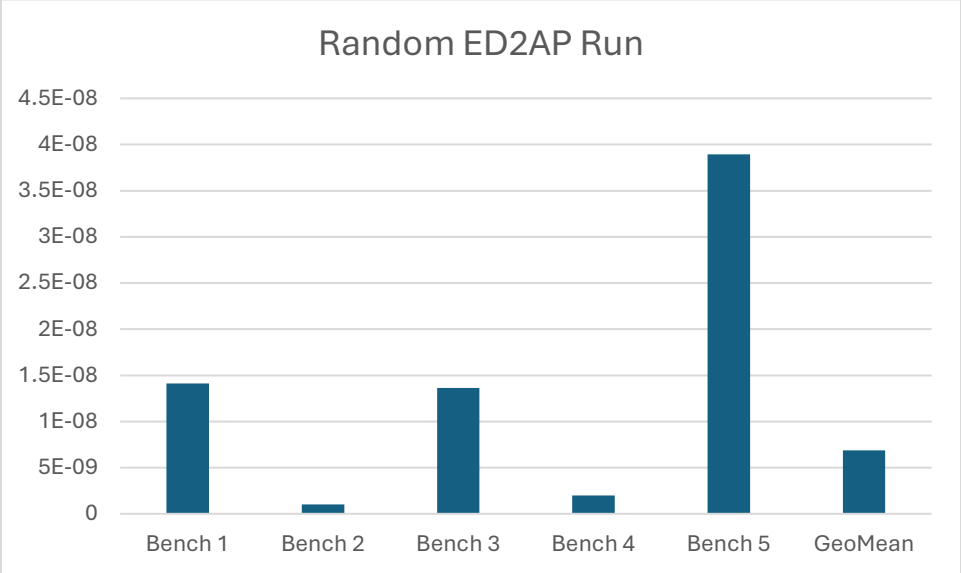
3.2 Bar Charts – Benchmark Normalized Metric at Best Design

Each bar chart shows the normalized values of the optimization metric across five benchmarks and the geomean for the best configuration found for each configuration.

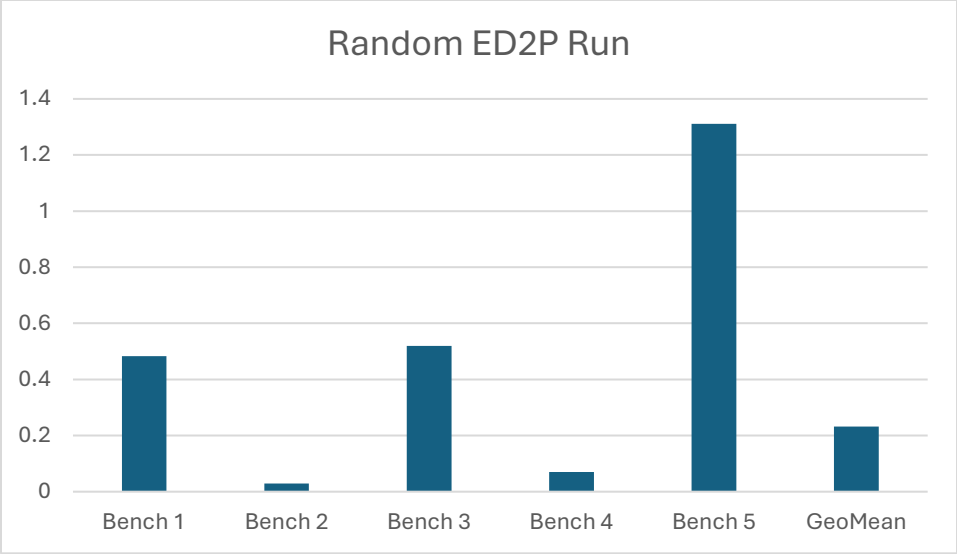


- EDP Bar Chart:– The geomean is ~ .55 meaning that this random configuration outperforms the baseline by 2x. Among the benchmarks, Bench 2 is the best-performing, with a normalized EDP ~.25 which implies that the random configuration worked exceptionally well for whatever workload that benchmark executes. On the other hand,

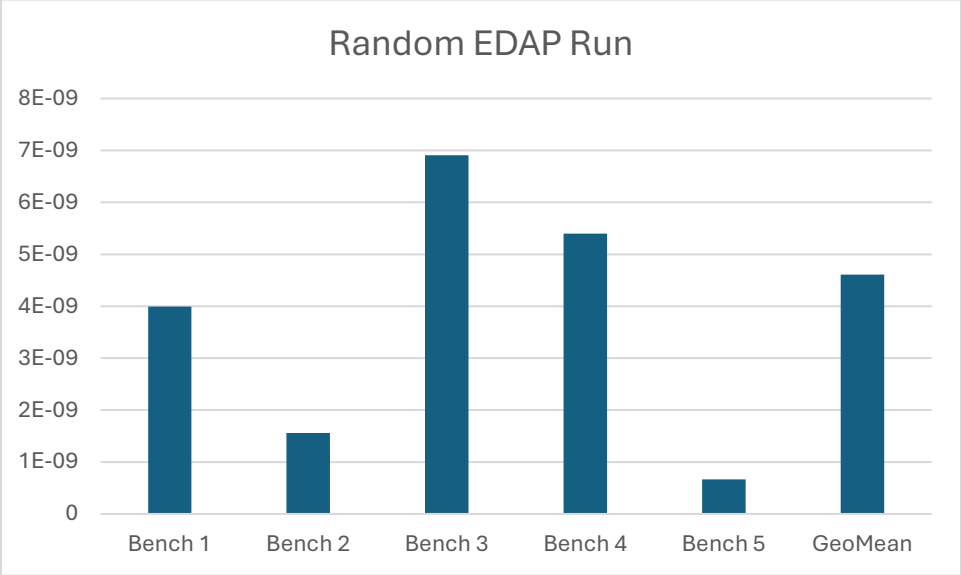
Bench 5 has the worst EDP with ~1.2. Compared to the line plot for EDP for the heuristic-based run, this runs best result is significantly worse. This chart displays that random search alone is unreliable for finding global optima in a massive design space because of this.



- ED^2AP Bar Chart: In this bar chart, all values fall within 10^{-9} to 10^{-8} showing that the configuration performs very well compared to the baseline. Benchmark 2 and Benchmark 4 have the lowest values, suggesting that this configurations is well-suited to workloads with modest delay and area demands in those tests. Benchmarks 1 and 3 performing well in terms of minimizing the combined energy, delay and area cost. Benchmark 6 most likely has a poor match between the configuration and benchmark's performance profile. This chart shows that while some benchmarks may benefit, others may suffer, which emphasizes the need for heuristics to consider architectural features with diverse benchmarks characteristics to account for a multitude of benchmarks.



- ED^2P Bar Chart: For this configuration, Benchmark 5 performs very poorly with an value exceeding 1.2 meaning it had a performance worse than the baseline, most likely due to a high delay component which is very punishing when squared. Benchmarks 1 and 4 are relatively similar and display moderate performance with values around 0.4. Benchmarks 2 and 4 performed exceptionally well as the configuration most likely matched these benchmarks architectural demands most likely due to smaller memory or branch behavior. The geometric mean was around 5 times better than the baseline with Benchmark 5 being the outlier in this run. The variability between benchmarks highlights the limitation fo using a random configuration on ED^2P as it is very sensitive to delay which means any mismatch can significantly impact the metric negatively.



- EDAP Bar Chart: All of the values fall within the 10^{-9} range, implying that the configuration performs significantly better than the baseline across all of the metrics. Benchmark 3 displays the highest EDAP value meaning that it was the least efficient for

this workload most likely due to cache hierarchy, buffer sizes, or area-heavy components that mismatched with the benchmark's behavior. Benchmarks 1 and 4 are relatively close to Bench 3 in value displaying a balanced yet imperfect performance across energy, delay, and area. Benchmark 2 and 5 perform significantly better than the others with implying that this configuration was well tuned for low latency, low area tasks within those benchmarks. Overall, this chart highlights that the configuration was competent and effective but not optimal, which shows why tailored heuristics are better.

4. Best Configurations

Each optimization produced a best configuration. For EDP, the optimal design had a width of 1, in-order execution, and a perfect predictor. It had very small L1 caches and a minimal L2 cache that satisfied the validation constraints. This configuration was compact and energy-efficient, and delay remained low due to the simplicity of the pipeline.

The best ED^2P configuration featured width 8, out-of-order execution, large buffers (RUU size 128 and LSQ size 32), and maximum L1 and L2 cache sizes. This design prioritized speed, and the squared delay metric made aggressive choices worth it even at high energy and area costs.

For EDAP, the best configuration avoided the perfect predictor and instead used a smaller prediction model. It had moderate cache sizes and in-order execution, balancing delay and energy while minimizing physical area.

The ED^2AP best configuration used a wide pipeline and out-of-order scheduling to avoid extreme delay penalties but traded off slightly on cache size and predictor complexity to reduce area overhead. This hybrid design represented a balanced approach tailored to the metric's needs.

5. Discussion

Each heuristic performed as expected, displaying expected behavior with respect to its optimization target. The EDP heuristic reached strong configurations very early, displaying that smaller setups are better in energy-centric scenarios. However, its simplicity is also a problem since it plateaued quickly, suggesting that there are few improvements to be made past a point. The ED^2P heuristic took longer to converge continued to improve across iterations. This most likely occurred because performance-optimized configurations benefit from more exploration of the higher resource portions of the design space. EDAP and ED^2AP heuristics performed more unpredictably as they suddenly improved after long periods of stability. This is most likely due to them making the space more complex thus requiring more experimentation to locate proper trade-offs.

A key takeaway is from this is how drastically different the configurations are based on the metric. For example, the perfect predictor is ideal for performance-based metrics but fails for area-sensitive metrics. Cache sizing also plays a large role in each configuration a it determines the latency and area. While the heuristics were effective they had their limitations. Each optimization only utilized one heuristic across 1000 iterations and it didn't learn from past iterations.

6. Conclusion

This project showed how custom heuristics can influence processor design space exploration when focusing on specific optimization goals. By limiting and focusing the search about configurations that concentrate on tradeoffs in energy, delay, and area, I was able to identify strong candidates for each of the target metrics. Considering how most of the designs through the iterations and their performances in the bar charts, we can better understand the patterns and impacts of our changes on our optimizations.