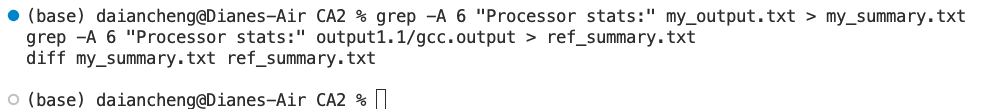
1. Introduction

This project implements a backend simulator for an out-of-order superscalar processor based on the Tomasulo algorithm. The simulator models dynamic instruction scheduling and resource conflicts across multiple pipeline stages: fetch, dispatch, scheduling, execution, and state update. Configurable parameters include fetch width, functional units (FUs), and result buses (R). The simulation produces cycle-by-cycle execution stats such as IPC and dispatch queue behavior. This report presents validation and design space analysis.

1. Validation

The simulator was tested against provided reference outputs using the diff utility. Results matched exactly for all benchmark traces (gcc, mcf, gobmk, hmmer), confirming correctness in pipeline timing and dependency handling.



A computer screen with white text

AI-generated content may be incorrect.A black text on a white background

AI-generated content may be incorrect.

1. Design Space Exploration

We varied key architectural parameters to observe their impact on IPC. Experiments covered:

1. Functional Units

Increasing FU count improves IPC, especially from minimal to moderate configurations (e.g., j1\_k1\_l1 → j2\_k1\_l1). Beyond that, gains diminish. Configurations like j2\_k1\_l2 often reach >95% of peak IPC, offering good performance-to-cost trade-offs.

1. Result Buses

IPC improves significantly as R increases from 1 to 4, but saturates afterward. R=4 consistently delivers >95% of maximum IPC across all traces, suggesting it eliminates the bus bottleneck.

1. Fetch Rate

Raising the fetch rate from F=4 to F=8 has negligible impact, indicating that fetch is not the limiting factor.

1. Minimal Configurations

Table 1 summarizes the minimal configurations that achieve ≥95% of peak IPC. For all benchmarks, R=4 is sufficient. Combined tuning of FUs and fetch rate could be explored in future work.

A graph with lines and numbers

AI-generated content may be incorrect.

Figure 1. IPC vs. Functional Unit configuration (j\_k\_l) across all traces.

A graph of a bus count

AI-generated content may be incorrect.

Figure 2. IPC vs. number of result buses (R) for fixed FU configuration.

A graph with colored lines

AI-generated content may be incorrect.

Figure 3. IPC vs. fetch rate (F) showing negligible change in performance.

A table with numbers and symbols

AI-generated content may be incorrect.

Table 1. Minimal result bus configurations (R) that achieve at least 95% of the peak IPC for each benchmark trace.

1. Conclusion

The simulator was validated across benchmarks and used to explore hardware-performance trade-offs. Key findings include: FUs impact IPC up to saturation;

result buses are a critical bottleneck; R=4 is generally sufficient; fetch rate has little impact beyond F=4. Recommended near-optimal configuration: FUs: j=2, k=1, l=2; Result Buses: R=4; Fetch Rate: F=4. This setup balances hardware efficiency and performance and serves as a practical baseline for superscalar backend design.