MCMASTER UNIVERSITY

FACULTY OF ENGINEERING

Department of Electrical and Computer Engineering

COMPENG 4DM4: Computer Architecture

Fall 2024

Project: OoO Execution Modeling (ROB and LSB)

Amr Elhossan, 400394120

Hydar Zartash, 400332624

Anthony Acosta, 400201942

Ahmad Hamzah, 400226804

Lab Partner Contribution

Team Member	Contributions
Hydar	Worked with Amr on the ROB design and code implementation + setup scripts to automate data acquisition
Amr	Worked with Hydar on the ROB design and code implementation + set up XML configuration files to operate with new ROB and LSB size, and IPC parameters + formatted and compiled the report.
Anthony	Worked with Ahmad on the LSB design and code implementation + setup scripts to automate data acquisition
Ahmad	Worked with Anthony on the LSB design and code implementation + setup scripts to automate data acquisition

Table 1: Lab Partners' Contributions

The project was done in a pair programming fashion, so contributions have significant overlap

Contents

1 Introduction

Modern processors achieve high performance by leveraging Out-of-Order (OoO) execution, an advanced optimization that maximizes instruction throughput. Unlike in-order execution, where instructions are processed sequentially, OoO pipelines dynamically schedule instructions based on the availability of operands and execution units. This approach allows the processor to exploit instruction-level parallelism (ILP), mitigating stalls caused by data dependencies and memory latency. By reordering execution while maintaining the illusion of sequential program order, OoO execution significantly enhances efficiency and performance in diverse workloads.

In this project, we analyze the behaviour of ROB and LSB under varying configurations. Using benchmarks, we explore the trends in execution cycles and LSB hits as a function of Instruction Per Cycle (IPC), ROB size, and LSB size.

2 Experimental Setup

2.1 Benchmarks

We evaluated the following benchmarks: `fdtd-apml`, `3mm`, `convolution-2d`, `doitgen`, `gemm`, `gemver`, `gramschmidt`, and `symm`.

2.2 Single-Core Cache Setup

The experiments were conducted using the following single-core cache configuration:

Table 2: Single-Core Cache Configuration

Parameter	Value
L1 Cache Size	8 KB
L1 Cache Associativity	1-way
L1 Cache Line Size	64 bytes
L1 Replacement Policy	RANDOM
L1 Latency	0 cycles
L2 Cache Size	32 KB
L2 Cache Associativity	2-way
L2 Cache Line Size	64 bytes
L2 Replacement Policy	LRU
L2 Latency	10 cycles
Main Memory Latency	250 cycles

2.3 Swept Parameters

The following parameters were swept during the experiments:

- **IPC (Instructions Per Cycle)**: Reflects the processor's ability to execute multiple instructions concurrently.
- **ROB (Reorder Buffer)**: Determines the maximum number of instructions that can be in-flight.
- **LSB (Load-Store Buffer)**: Represents the size of the memory operations buffer.

2.4 Metrics

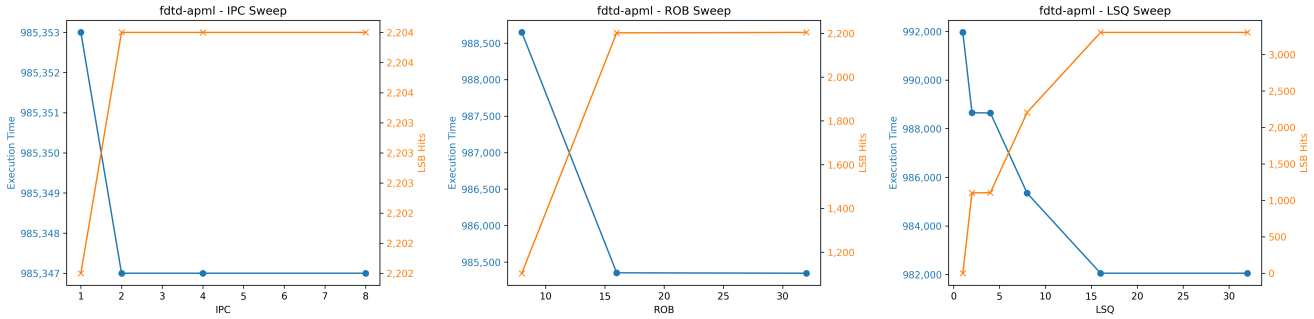
For each parameter sweep, we measured:

- **Execution Time**: Total cycles to complete the benchmark.
- **LSB Hits**: The number of load forward operations handled by the Load-Store Buffer.

3 Results and Analysis

The impact of ROB, IPC, and LSQ varies across benchmarks, as performance is inherently tied to the characteristics of the program being executed, as discussed in class.

3.1 `fdtd-apml`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

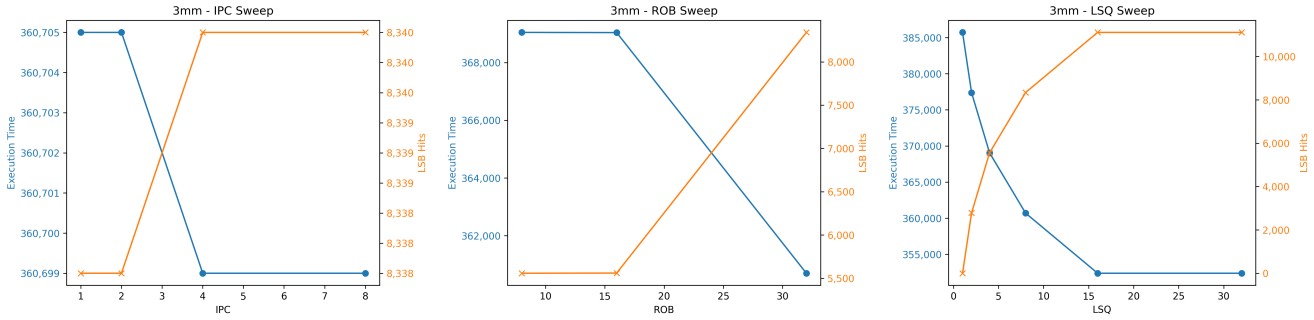
Figure 1: `fdtd-apml` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `fdtd-apml` sweeps are as follows:

- **IPC Sweep:** Execution time decreases significantly for IPC values up to 2, with diminishing returns beyond this point. LSB hits increase slightly for IPC values up to 2.
- **ROB Sweep:** Increasing ROB size reduces execution time initially and increases LSB hits, as more instructions can be in-flight. Performance plateaus beyond a certain ROB size.
- **LSB Sweep:** Larger LSB sizes resulted in reduced execution cycles and stable LSB hits after saturation.

3.2 `3mm`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

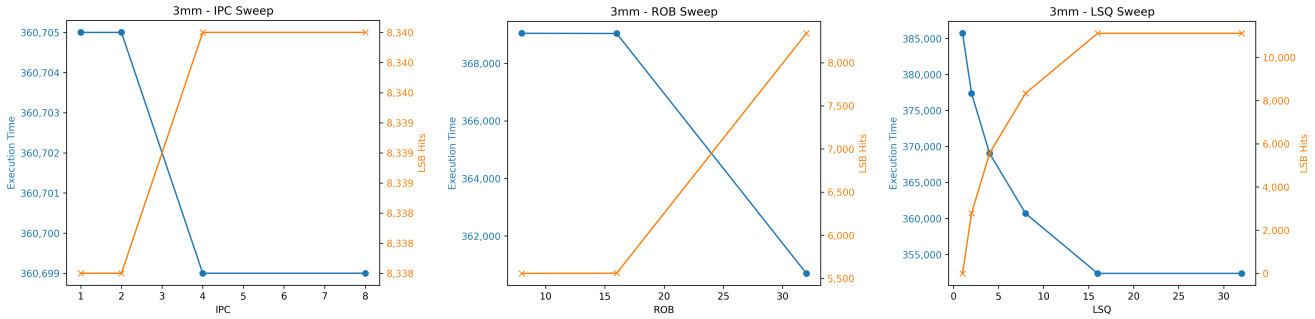
Figure 2: `3mm` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `3mm` sweeps are as follows:

- **IPC Sweep:** Execution time decreases sharply with increasing IPC, tapering off at higher IPC values. LSB hits grow marginally with IPC.
- **ROB Sweep:** Execution time reduces steadily with increasing ROB size, stabilizing after reaching a threshold. LSB hits show consistent growth with ROB size.
- **LSB Sweep:** Execution time improves with larger LSB sizes, while LSB hits plateau beyond a certain point.

3.3 `convolution-2d`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

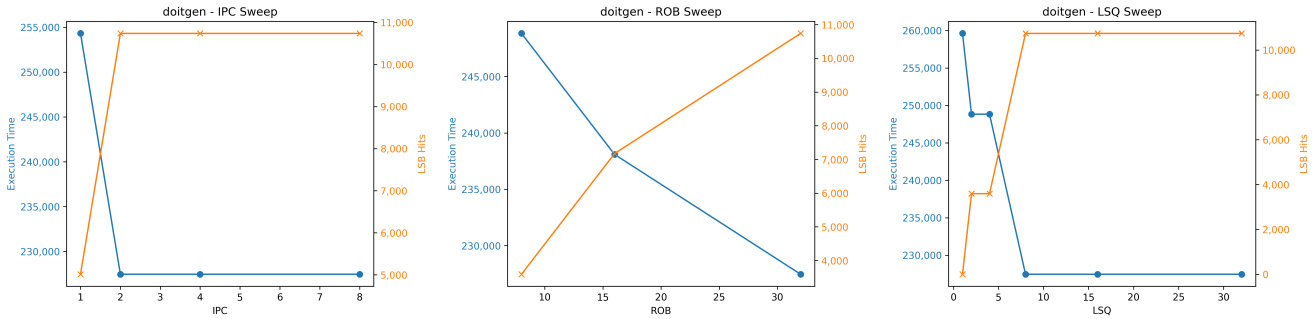
Figure 3: `convolution-2d` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `convolution-2d` sweeps are as follows:

- **IPC Sweep:** Execution time decreases sharply for lower IPC values and tapers off at higher IPC values. LSB hits show minimal variation.
- **ROB Sweep:** Execution time decreases with increasing ROB size, eventually stabilizing. LSB hits show consistent growth with ROB size perhaps tapering off at a larger size.
- **LSB Sweep:** Execution time decreases as LSB size increases, with LSB hits rise and stabilize after saturation.

3.4 `doitgen`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

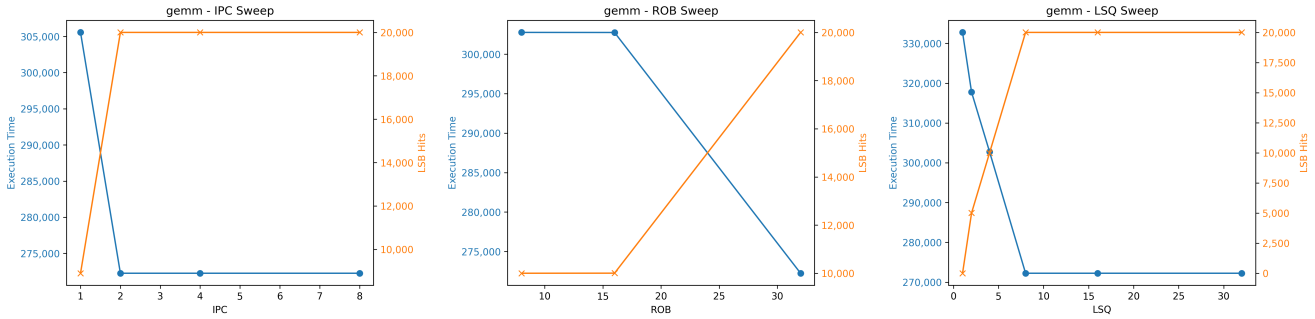
Figure 4: `doitgen` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `doitgen` sweeps are as follows:

- **IPC Sweep:** Execution time decreases sharply as IPC increases, with diminishing returns at higher IPC values. LSB hits increase slightly.
- **ROB Sweep:** Execution time reduces as ROB size increases, leveling off beyond a threshold. LSB hits increase steadily appearing to taper off.
- **LSB Sweep:** Execution time improves significantly with larger LSB sizes, while LSB hits grow in a step fashion and stabilize after a certain point.

3.5 `gemm`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

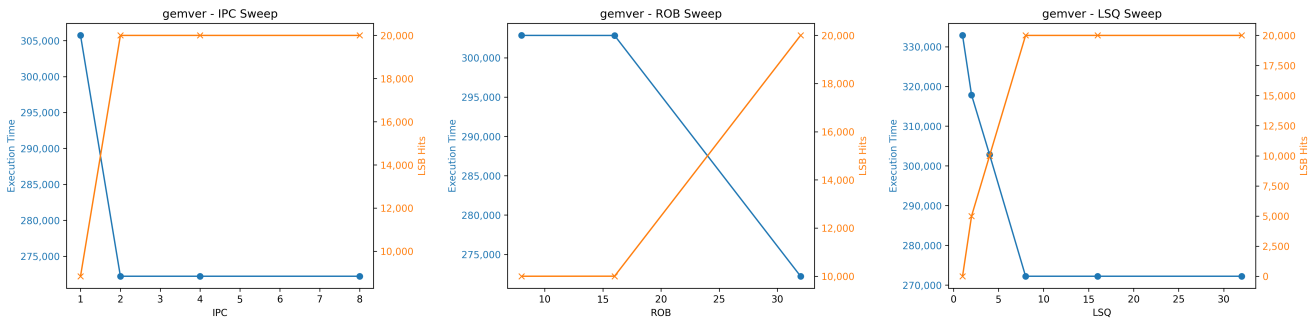
Figure 5: `gemm` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `gemm` sweeps are as follows:

- **IPC Sweep:** Execution time decreases noticeably as IPC increases, with diminishing returns at higher IPC values. LSB hits exhibit large growth.
- **ROB Sweep:** Execution time decreases steadily with increasing ROB size and stabilizes beyond a threshold. LSB hits grow proportionally with ROB size perhaps tapering off at a larger size.
- **LSB Sweep:** Execution time decreases with larger LSB sizes hitting a floor, while LSB hits increase and stabilize after reaching saturation.

3.6 `gemver`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

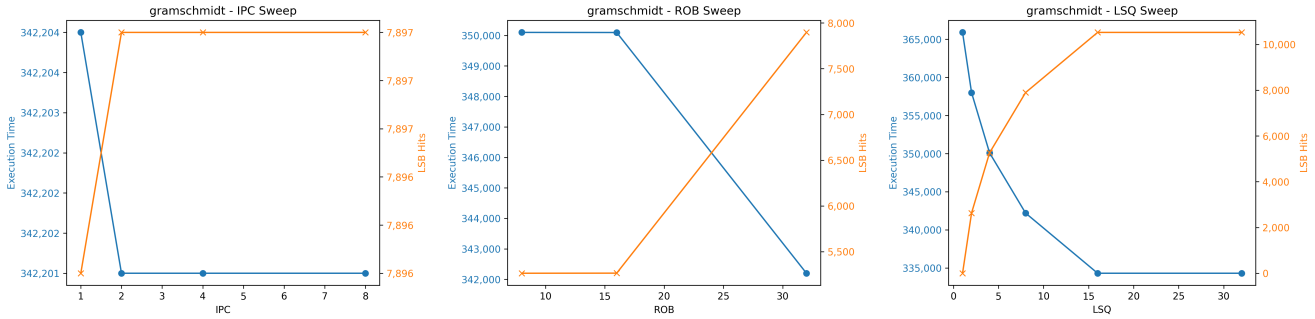
Figure 6: `gemmver` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `gemmver` sweeps are as follows:

- **IPC Sweep:** Execution time decreases with increasing IPC, with marginal improvements at higher IPC values. LSB hits grow fast and hit a ceiling.
- **ROB Sweep:** Execution time reduces progressively with increasing ROB size, eventually plateauing. LSB hits show gradual growth.
- **LSB Sweep:** Execution time decreases as LSB size increases, with LSB hits stabilizing beyond saturation.

3.7 `gramschmidt`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

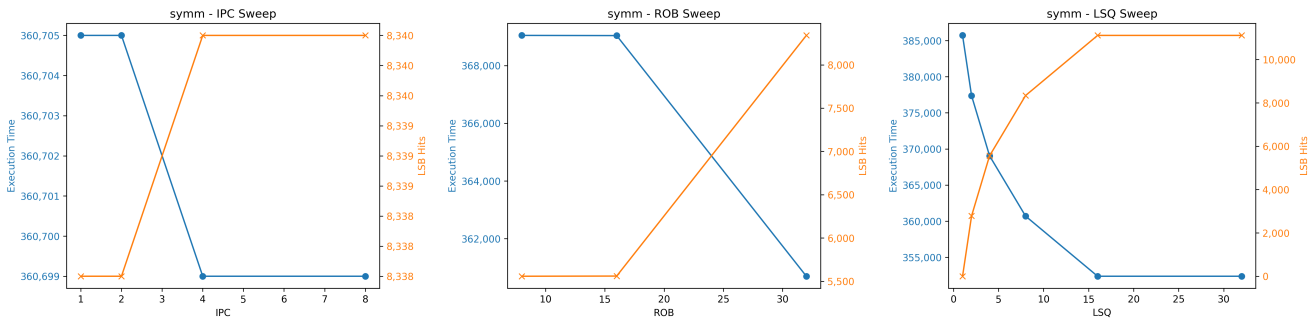
Figure 7: `gramschmidt` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `gramschmidt` sweeps are as follows:

- **IPC Sweep:** Execution time decreases significantly for IPC values up to 2, with diminishing returns beyond this point. LSB hits increase slightly for IPC values up to 2.
- **ROB Sweep:** Increasing ROB size reduces execution time initially and increases LSB hits, as more instructions can be in-flight. Performance plateaus beyond a certain ROB size.
- **LSB Sweep:** Larger LSB sizes resulted in reduced execution cycles and stable LSB hits after saturation.

3.8 `symm`



(a) IPC Sweep (b) ROB Sweep (c) LSB Sweep

Figure 8: `symm` Sweeps: IPC, ROB, and LSB trends.

Analysis

The trends observed in `symm` sweeps are as follows:

- **IPC Sweep:** Execution time decreases significantly for IPC values up to 2, with diminishing returns beyond this point. LSB hits increase slightly for IPC values up to 2.
- **ROB Sweep:** Increasing ROB size reduces execution time initially and increases LSB hits, as more instructions can be in-flight. Performance plateaus beyond a certain ROB size.
- **LSB Sweep:** Larger LSB sizes resulted in reduced execution cycles and stable LSB hits after saturation.

3.9 General Observations and Explanations

For all the benchmarks, the following trends were generally observed:

- **Increasing IPC decreases execution time and increases LSB hits:** Higher IPC allows more instructions to be retired per cycle, allowing memory instructions to arrive and/or be served sooner. Since only the first memory request at the front of the LSB is sent to the memory controller, this creates a bottleneck. Despite this, the increased rate of memory request arrivals result in slightly more LSB hits due to the frequent load forwarding from the buffered stores.
- **Increasing ROB size decreases execution time and increases LSB hits:** A larger ROB allows the simulator to manage more in-flight instructions, ensuring a steady supply of memory operations to the LSB. This reduces execution time as more instructions progress toward retirement. The increased memory activity contributes to higher LSB hits, even though only one memory request is served at a time.
- **Increasing LSB size decreases execution time and increases LSB hits:** A larger LSB provides greater capacity to buffer memory operations, reducing stalls caused by insufficient buffer space. While only the oldest request is served, the increased buffer size enables more load forwarding opportunities of queued requests, leading to higher LSB hits and reduced execution time.

4 Conclusion

This project demonstrated the impact of ROB and LSB configurations on processor performance across various benchmarks. Our findings align with theoretical expectations, highlighting the significance of optimizing these parameters for efficient OoO execution. Future work could explore implementing a more real-word simulator that involves COMPUTE dependencies.