Out-of-Order Superscalar Execution: Resolving Data Hazards in MIPS Architecture

Subrahmanya Rao D P
Dept. of ECE
RV College of Engineering Bangaluru, India
Subrahmanyardp.ec21@rvce.edu.in

Shravankumar N Biradar
Dept. of ECE
RV College of Engineering Bangaluru, India shravankumarnb.ec21@rvce.edu.in

Aishwarya H
Dept. of ECE
RV College of Engineering
Bangaluru, India
aishwaryah.ec21@rvce.edu.in

Charan H A
Dept. of ECE
RV College of Engineering Bangaluru, India
charanha.ec21@rvce.edu.in

Dr. Jayanthi P N
Dept. of ECE
RV College of Engineering Bangaluru, India jayanthipn@rvce.edu.in

***Abstract*—Out-of-order superscalar execution is a critical advancement in modern processor architectures, enhancing instruction-level parallelism (ILP) and mitigating performance bottlenecks. This paper explores the implementation of superscalar execution in MIPS architecture, focusing on resolving data hazards such as Read-After-Write (RAW), Write-After-Read (WAR), and Write-After-Write (WAW). Techniques like Tomasulo's algorithm, register renaming, reservation stations, and branch prediction are employed to optimize execution flow. A Python-based simulation demonstrates improved throughput, reduced stalls, and efficient CPU resource utilization. The results highlight the impact of dynamic scheduling and speculative execution in maximizing computational efficiency, contributing to the evolution of high-performance processor designs.**

I. Introduction

Modern processor architectures aim to maximize performance by efficiently executing multiple instructions in parallel. MIPS (Microprocessor without Interlocked Pipeline Stages) is a RISC (Reduced Instruction Set Computing) architecture known for its simplicity and high-performance pipeline design. Traditionally, MIPS processors execute instructions in-order, meaning they follow a strict sequential execution model. However, as computational demands increase, in-order execution becomes a limiting factor due to pipeline stalls caused by data, control, and structural hazards. These hazards prevent instructions from executing efficiently, reducing overall throughput.

To address these challenges, modern processors employ superscalar execution, which allows multiple instructions to be issued and executed per cycle, leveraging multiple functional units. Additionally, out-of-order execution (OoO) enhances performance by dynamically scheduling instructions based on operand availability rather than program order, reducing pipeline stalls and improving resource utilization.

The primary bottleneck in executing instructions efficiently is data hazards, where an instruction depends on the result of a previous instruction that has not yet completed. To mitigate these hazards, advanced techniques such as Tomasulo's algorithm, register renaming, reservation stations, and branch prediction are implemented. These techniques ensure that independent instructions execute without unnecessary delays, thereby optimizing CPU efficiency.

This paper explores the implementation of out-of-order superscalar execution in MIPS, focusing on resolving data hazards. A Python-based simulation is used to model instruction execution, demonstrating how dynamic scheduling and speculative execution improve instruction throughput. The findings provide insights into how superscalar architectures drive high-performance computing by minimizing stalls and maximizing parallelism.

II. Litrature Survey

The paper titled "Speculative Execution in Modern Superscalar Processors: Opportunities and Challenges" (ACM Computing Surveys, 2019) explores the role of speculative execution in enhancing instruction-level parallelism (ILP) and processor efficiency while addressing its security implications. It discusses key techniques such as branch prediction, out-of-order execution, and memory speculation, which improve performance by reducing stalls and optimizing resource utilization. Real-world implementations in architectures like Intel Core, AMD Ryzen, and Apple M1 demonstrate significant throughput improvements. However, the paper also highlights security vulnerabilities, notably Spectre and Meltdown attacks, which exploit speculative execution to access sensitive data. To mitigate these risks, researchers propose hardware-based solutions like SafeSpec and software-level fixes such as Speculative Execution Barriers (SEBs). The study also explores AI-driven speculative execution models to enhance prediction accuracy while minimizing security threats. Additionally, it examines the trade-offs between performance gains and security risks, emphasizing the need for secure speculation mecha-

Identify applicable funding agency here. If none, delete this.

nisms. Future research focuses on speculation-aware compiler optimizations, enhanced hardware validation techniques, and quantum computing-inspired execution models, paving the way for safer and more efficient speculative execution in modern superscalar processors.

The paper titled ”Dynamic Scheduling in High-Performance Superscalar Processors” explores dynamic instruction scheduling as a key technique to enhance instruction-level parallelism (ILP) and processor efficiency. Unlike traditional in-order execution, dynamic scheduling enables out-of-order execution, allowing instructions to execute as soon as their operands are available, reducing pipeline stalls caused by data dependencies, control hazards, and resource conflicts. The study examines scoreboarding and Tomasulo’s algorithm, where scoreboarding tracks dependencies to optimize functional unit utilization, while Tomasulo’s algorithm introduces register renaming to eliminate false dependencies (WAR and WAW hazards), ensuring higher parallel execution. These techniques maximize execution unit efficiency in superscalar processors, significantly improving throughput. However, the paper also discusses the hardware complexity trade-offs, as dynamic scheduling requires additional structures like reservation stations and reorder buffers (ROB), increasing design complexity. Furthermore, challenges such as branch mispredictions and memory access latencies necessitate advanced prediction and speculation techniques for further optimization. Despite these challenges, dynamic scheduling revolutionized superscalar processor design, paving the way for modern high-performance architectures such as those in Intel, AMD, and ARM processors, where out-of-order execution, speculative execution, and efficient resource utilization are fundamental to achieving higher computational efficiency.

III. Methodology

The implementation of out-of-order superscalar execution in MIPS architecture follows a structured and systematic approach to enhance instruction-level parallelism (ILP) and minimize pipeline stalls caused by data, control, and structural hazards. The methodology involves simulation-based modeling using Python and Jupyter Notebook, allowing for a detailed analysis of superscalar execution and dynamic instruction scheduling in a MIPS-based pipeline. The approach consists of multiple stages, including hazard identification, pipeline design, dynamic scheduling implementation, hazard resolution, and performance evaluation.

A. Understanding Data Hazards and Dependencies

The first step involves analyzing instruction dependencies in a MIPS pipeline to identify RAW (Read-After-Write), WAR (Write-After-Read), and WAW (Write-After-Write) hazards. These hazards create execution bottlenecks, preventing instructions from progressing efficiently through the pipeline. By understanding these dependencies, strategies can be formulated to resolve hazards dynamically and improve throughput.

B. Design and Implementation of Superscalar Execution

To enhance execution efficiency, a superscalar pipeline model is developed in Python, allowing multiple instructions to be fetched, decoded, and executed in parallel. Unlike traditional scalar pipelines, which execute only one instruction per cycle, superscalar execution enables multiple functional units to process instructions simultaneously. This step involves designing:

- Instruction Fetch and Decode Stages that can handle multiple instructions per cycle.
- Multiple Execution Units for parallel processing of arithmetic, memory, and branch instructions.
- Instruction Dispatch Mechanisms to efficiently allocate operations to available functional units.

C. Out-of-Order Execution and Dynamic Scheduling

A key challenge in superscalar execution is ensuring efficient instruction scheduling to avoid stalls. To address this, Tomasulo’s Algorithm is implemented to facilitate out-of-order execution, allowing instructions to execute as soon as their operands are available rather than strictly following program order. Additionally:

- Register renaming is incorporated to eliminate false dependencies (WAR and WAW hazards), ensuring that different instructions do not interfere with one another.
- A reservation station mechanism is used to dynamically schedule instructions and allocate execution units efficiently.
- A Reorder Buffer (ROB) ensures instructions commit in order, preserving program correctness and handling exceptions precisely.

D. Hazard Detection and Resolution Techniques

Pipeline hazards can significantly impact processor performance by causing stalls and reducing instruction throughput. To mitigate these issues, several hazard detection and resolution techniques are implemented in the out-of-order superscalar MIPS execution model. RAW (Read-After-Write) hazards are addressed using operand forwarding, which allows data to be directly passed from one instruction to the next without waiting for it to be written back to the register file. This technique significantly reduces unnecessary delays and ensures that dependent instructions can execute without stalling the pipeline.

Additionally, branch prediction and speculative execution are incorporated to tackle control hazards. A dynamic branch predictor forecasts the outcome of conditional branches, allowing the processor to speculatively fetch and execute instructions. This reduces pipeline stalls caused by branch delays. If the prediction is correct, execution continues seamlessly; if incorrect, replay mechanisms ensure that mispredicted instructions are discarded and the correct execution path is followed. To handle WAR (Write-After-Read) and WAW (Write-After-Write) hazards, register renaming is employed, which eliminates false dependencies by dynamically mapping

logical registers to physical registers. These techniques collectively enhance parallel execution, minimize pipeline stalls, and improve overall processor efficiency, ensuring that the superscalar MIPS pipeline can process multiple instructions simultaneously without unnecessary interruptions.

E. Simulation and Performance Analysis

To evaluate the effectiveness of the implemented out-of-order superscalar execution model, a Python-based simulation is conducted using a set of sample MIPS instructions. The simulation models how multiple instructions move through the pipeline, demonstrating how dynamic scheduling, register renaming, and hazard resolution improve execution efficiency. Key performance metrics such as instruction throughput, stall reduction, and execution time improvement are analyzed. The results show that out-of-order execution allows independent instructions to progress without waiting for previous instructions to complete, significantly reducing pipeline stalls.

Furthermore, the impact of branch prediction and speculative execution is examined. Accurate branch prediction reduces the number of mispredictions and unnecessary instruction flushes, ensuring a smoother execution flow. Speculative execution further improves parallelism by executing instructions before confirming their necessity, boosting overall performance. The efficiency of these techniques is measured by comparing execution cycles with and without speculation. The simulation results highlight that superscalar processors with out-of-order execution achieve higher instruction throughput and better resource utilization compared to traditional in-order pipelines. These findings confirm the effectiveness of dynamic instruction scheduling, register renaming, and branch prediction in optimizing modern MIPS-based architectures, reinforcing their importance in high-performance computing.

RESULTS AND DISCUSSIONS

The implementation of out-of-order superscalar execution in MIPS architecture was successfully simulated using a Python-based model, demonstrating the impact of dynamic scheduling, register renaming, and speculative execution on processor performance. The results indicate a significant improvement in instruction throughput due to the ability of the processor to execute multiple instructions per cycle, efficiently utilizing available execution units.

A key observation was the reduction in pipeline stalls, particularly those caused by data hazards (RAW, WAR, and WAW). The implementation of Tomasulo's Algorithm and register renaming ensured that instructions could execute without waiting for previous operations to complete, thereby eliminating unnecessary dependencies. Additionally, operand forwarding proved effective in handling RAW hazards, allowing dependent instructions to proceed without delays.

The simulation also highlighted the benefits of branch prediction and speculative execution. By accurately predicting branch outcomes, the pipeline was able to fetch and execute instructions in advance, reducing the number of cycles lost to mispredictions. In cases where incorrect predictions occurred, replay mechanisms efficiently recovered from errors, ensuring program correctness. The use of a reorder buffer (ROB) further maintained in-order instruction commitment, preventing inconsistencies in execution flow.

Performance analysis showed a notable increase in execution speed compared to traditional in-order pipelines. The processor achieved higher parallelism, resulting in lower execution latency and improved resource utilization. These results confirm that superscalar and out-of-order execution techniques are essential for high-performance computing, enabling modern processors to handle complex workloads efficiently. The study reinforces the importance of advanced instruction scheduling and hazard resolution techniques in optimizing MIPS-based architectures.

FUTURE SCOPE

The implementation of out-of-order superscalar execution in MIPS architecture lays the foundation for further advancements in high-performance computing. One key area of future research is the enhancement of dynamic scheduling techniques using AI-driven instruction scheduling. Machine learning models can predict instruction dependencies and execution patterns more efficiently than traditional approaches, optimizing out-of-order execution while reducing stalls. Additionally, adaptive branch prediction powered by AI could significantly improve prediction accuracy, minimizing the performance impact of mispredictions. Another crucial area is security enhancements in speculative execution to mitigate vulnerabilities like Spectre and Meltdown. Hardware-based solutions and secure speculation techniques can help balance performance and security, ensuring safe execution without compromising efficiency.

Further research could explore multi-threaded superscalar execution through simultaneous multithreading (SMT), allowing multiple instruction streams to be executed in parallel within a single core, enhancing parallelism and computational throughput. The adoption of 3D-stacked processors and chiplet-based architectures can further improve scalability and interconnect efficiency, leading to higher data processing speeds. Additionally, power-efficient out-of-order execution methods can be developed to reduce energy consumption while maintaining performance, making these architectures more suitable for embedded systems and battery-powered devices. Expanding this work into heterogeneous computing, such as integrating GPUs and AI accelerators, could further optimize performance for machine learning, deep learning, and real-time applications. These advancements will drive the evolution of next-generation superscalar processors, making them faster, more energy-efficient, and adaptable to future computing demands.

CONCLUSIONS

The implementation of out-of-order superscalar execution in MIPS architecture demonstrates the effectiveness of dynamic instruction scheduling, register renaming, and speculative execution in improving instruction-level parallelism (ILP) and overall processor efficiency. By enabling multiple instructions

per cycle and allowing instructions to execute as soon as their operands are available, superscalar execution significantly reduces pipeline stalls caused by data, control, and structural hazards. The integration of Tomasulo's Algorithm, operand forwarding, branch prediction, and reorder buffers (ROB) ensures smooth execution, optimizing CPU resource utilization. The simulation results confirm that superscalar and out-of-order execution techniques lead to higher instruction throughput and lower execution latency, proving their importance in modern high-performance computing.

While the study successfully highlights the advantages of superscalar execution in MIPS, challenges such as hardware complexity, power consumption, and security risks in speculative execution remain areas for further improvement. Future advancements in AI-driven scheduling, energy-efficient execution models, and secure speculation techniques will continue to refine these architectures. The findings reinforce the significance of superscalar execution and out-of-order processing in modern processor design, paving the way for more efficient, scalable, and high-performance computing architectures in the future.

REFERENCES

[1] Out-of-order execution may not be cost-effective on processors featuring simultaneous multithreading, January 1999.
[2] An Out-of-Order Superscalar Processor Using STRAIGHT Architecture in 28 nm CMOS, July 2023
[3] In-N-Out: Reproducing Out-of-Order Superscalar Processor Behavior from Reduced In-Order Traces, August 2011.