Register Renaming

6.375 Final Project Presentation

Abdulsami Aldahlawi

Khalid Al-Hawaj

Agenda

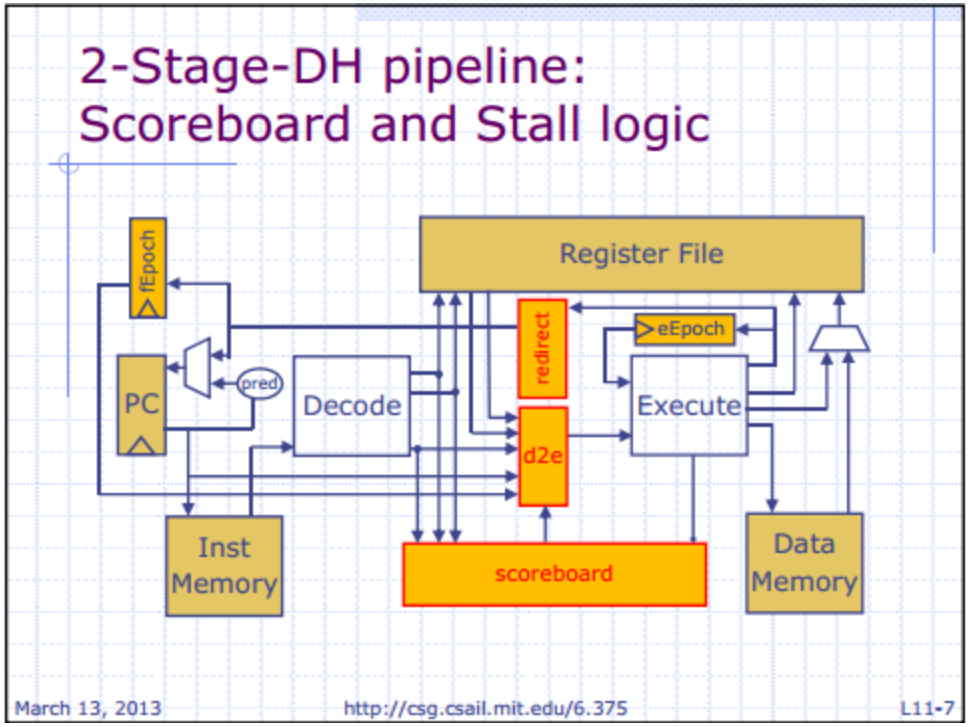
- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Problem Description

- Recap on scoreboard for data hazard handling …



- Problem: unnecessary stalls for WAW hazards.

Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Background

- Compiler Scheduling (static) vs CPU Scheduling (dynamic)
  - ISA limitation
  - More information available in runtime

- Example:
```
LD   R1, 0(R5)
ADD  R1, R2, R3
```

- Using scoreboard, we have to stall …
- Solution: reassign destination registers in runtime

```
LD   PR32, 0(PR13)
ADD  PR54, PR22, PR33
```

Background – How to Rename ?

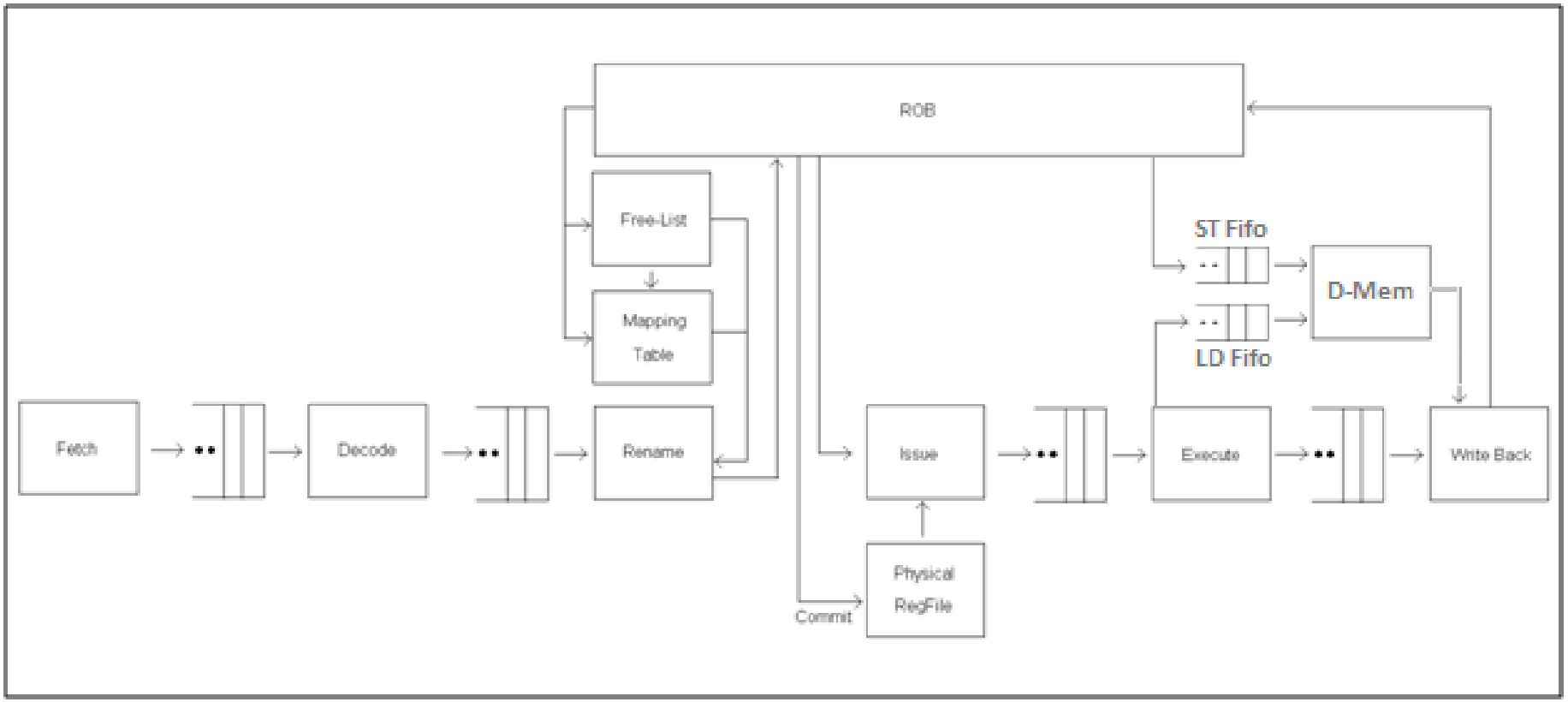
- Tomasulo's Algorithm (e.g. Intel-P6 style)
  - Use of multiple distributed Reservation Station and a Common Data Bus
  - Issues: - Bus is expensive
    - Infeasible for modern systems

- Sohi's Method (e.g. MIPS R10K)
  - Separate Logical RegFile (defined by ISA) from Physical RegFile.
  - Maintain Mapping Table and Free List.

Agenda

- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Processor Overview



- Fetch, Decode, Execute, and WriteBack are almost the same as the SMIPS we implemented in the Lab.

Agenda

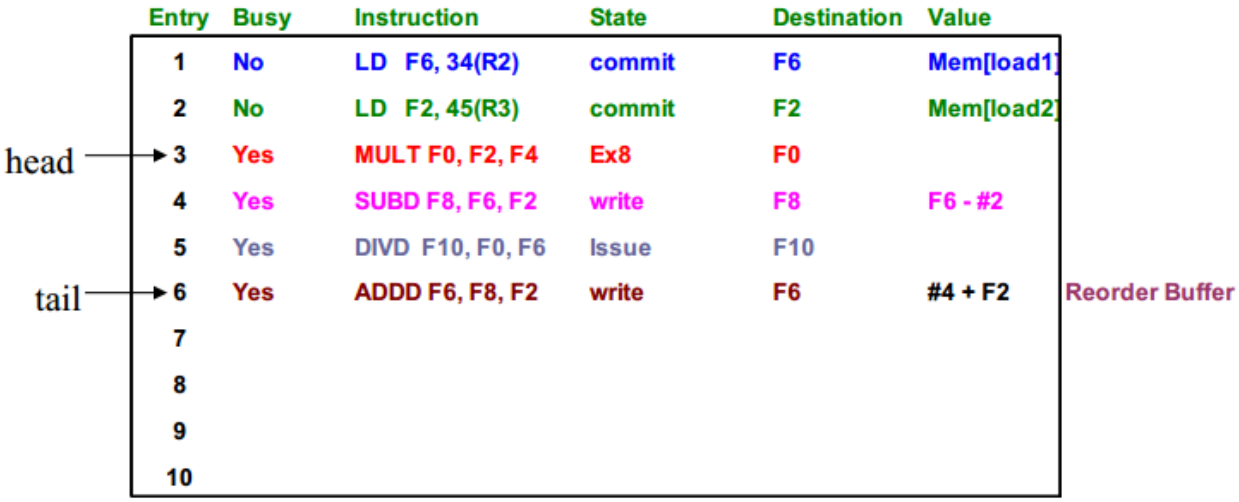
- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Microarchitectural Description

- Rename:
  - Takes ADD R1, R2, R3 as input
  - Consult 'Mapping Table' (holds the current mapping of Logical registers to Physical registers) to get physical registers corresponding to sources R2 and R3. (e.g. PR15 and PR18).
  - Assigns a new physical destination register to R1 from the 'Free List' and change its previous mapping in 'Mapping Table'.

Microarchitectural Description

- Re-Order Buffer:
  - Holds renamed instructions and their status.
  - Keeps track of instruction flow in the pipeline.



- Commit:
  - Evict the head of the ROB when its results has been written back
  - Frees the previous mapping of the physical register file.

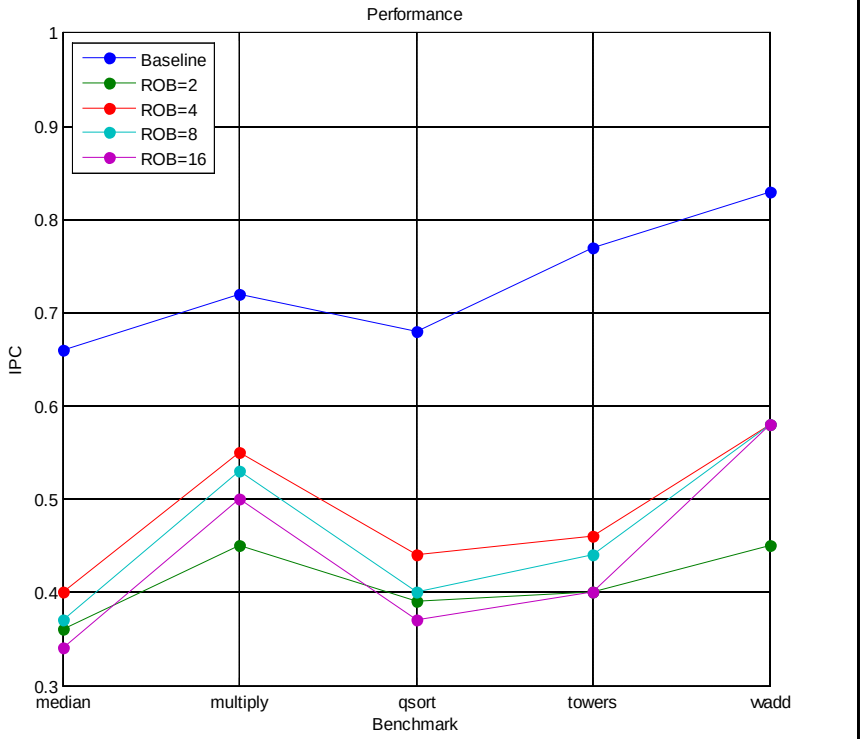
Agenda

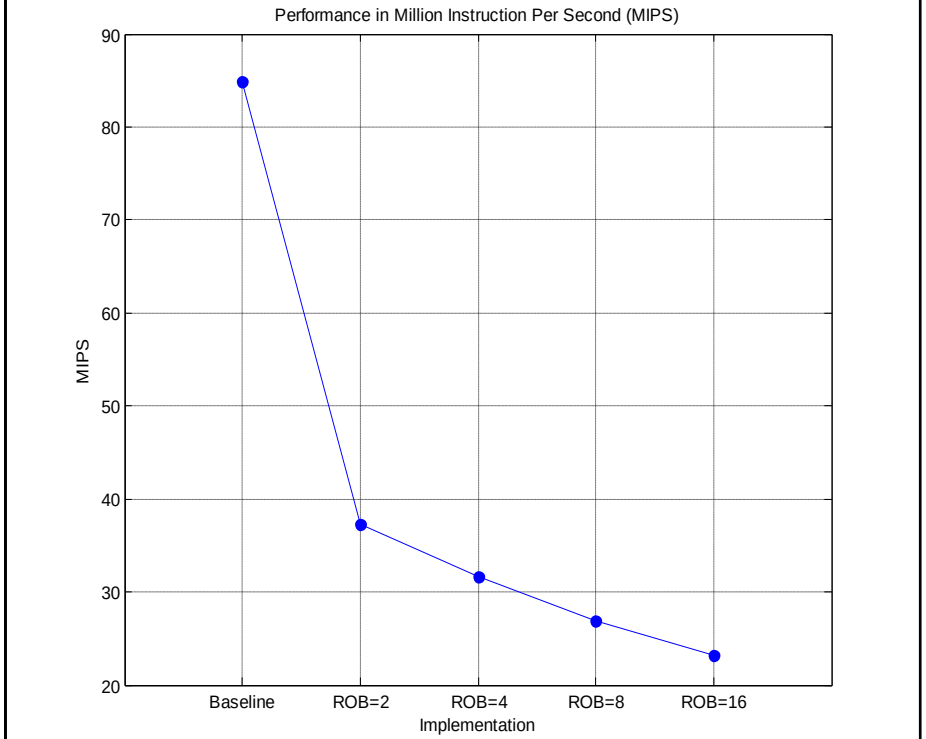
- Problem Description
- Background
- Processor Overview
- Microarchitectural Description
- Simulation and FPGA Results

Simulation and FPGA Results

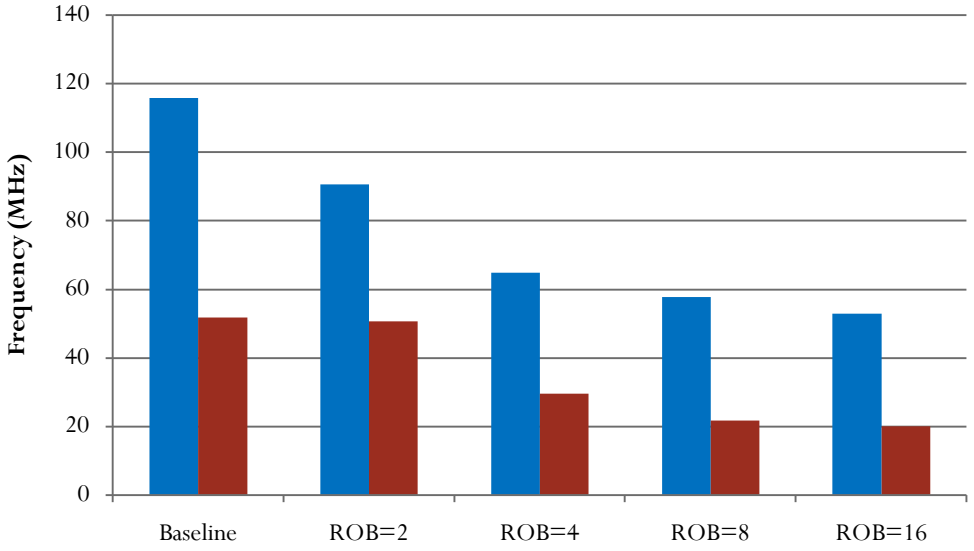
- Comparison in terms of performance (IPC and MIPS), Maximum achievable frequency, and Device Utilization were made on the following implementations:
  - Baseline: Basic 5-Stage in-order processor (Lab version)
  - Renaming: ROB Size = 2, RegFile Size = 32
  - Renaming: ROB Size = 4, RegFile Size = 64
  - Renaming: ROB Size = 8, RegFile Size = 64
  - Renaming: ROB Size = 16, RegFile Size = 64

Simulation and FPGA Results

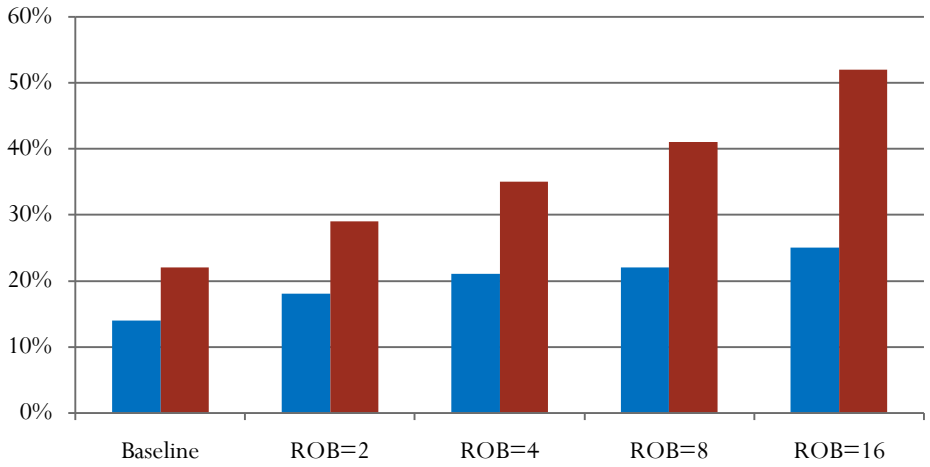




Simulation and FPGA Results



Maximum Frequency



Device Utilization

Questions

References

- [1] Tomasulo, R. M., "An Efficient Algorithm for Exploiting Multiple Arithmetic Units",Sept. 1965.
- [2] Brooks, David., “Tomasulo’s Algorithm”, CS246: Advanced Computer Architecture – Harvard School of Engineering and Applied Sciences Course, Feb. 2013.
- [3] Brooks, David., “MOB, P6 and R10K”, CS246: Advanced Computer Architecture – Harvard School of Engineering and Applied Sciences Course, Feb. 2013.