MS108
Computer System(1)
Course Project

My Journey with RISCV-CPU

December 24 2018

Bohan Hou

Overview

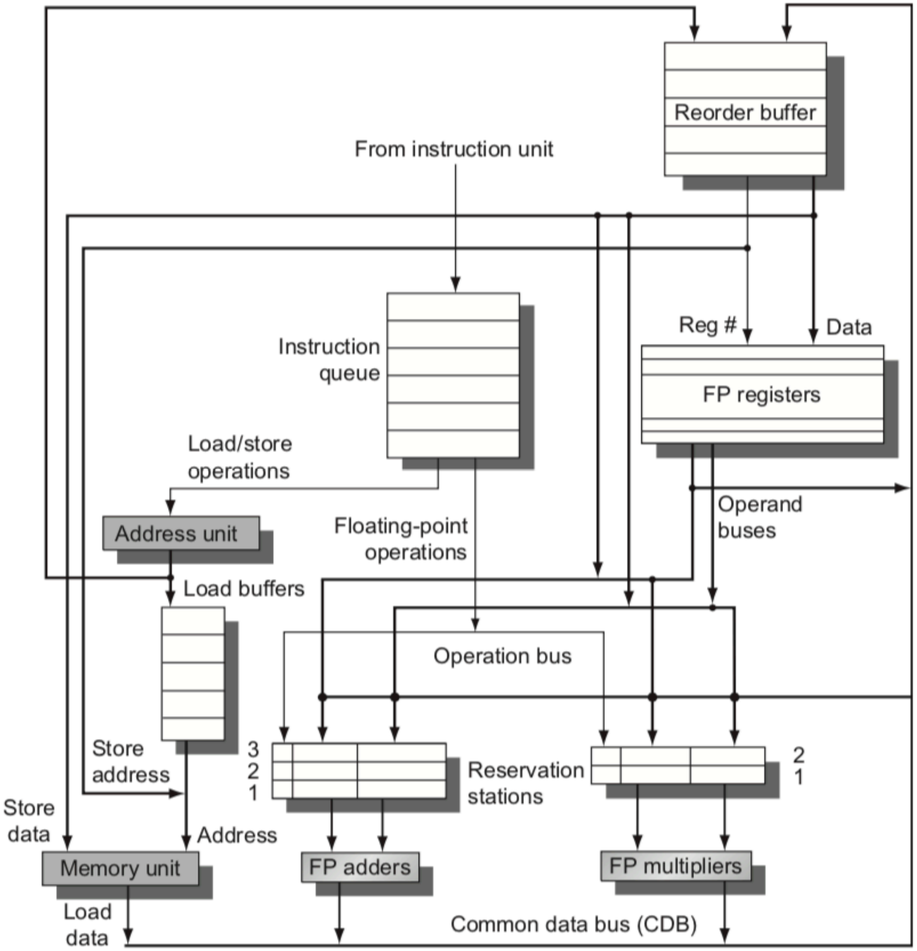
- What I’ve done
- What I’ve learned

Prelude & Tomasulo Algorithm

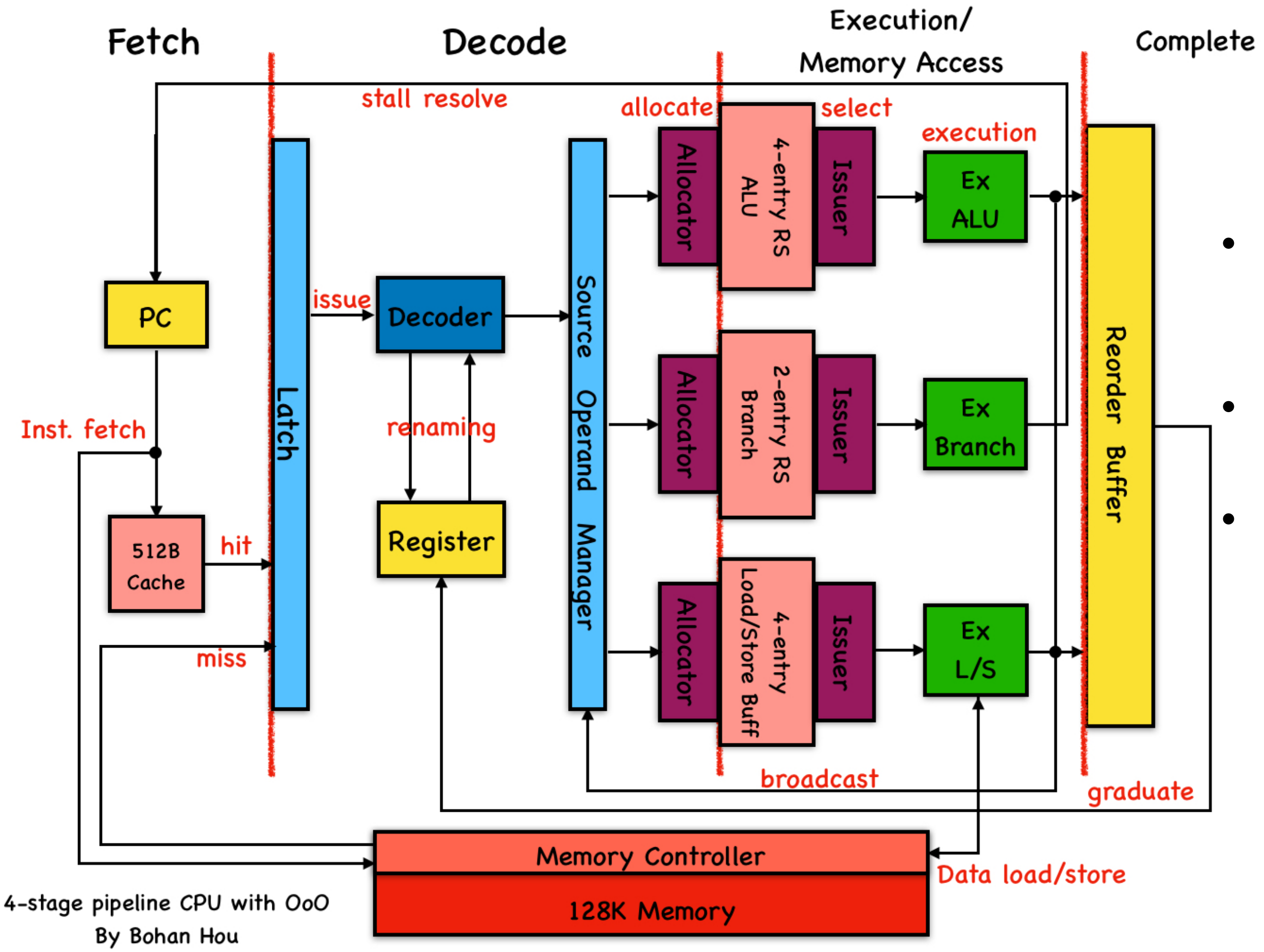
- Elegant design

- More a modern processor microarchitecture than standard 5-stage pipeline

- Put theory into practice and see what happens



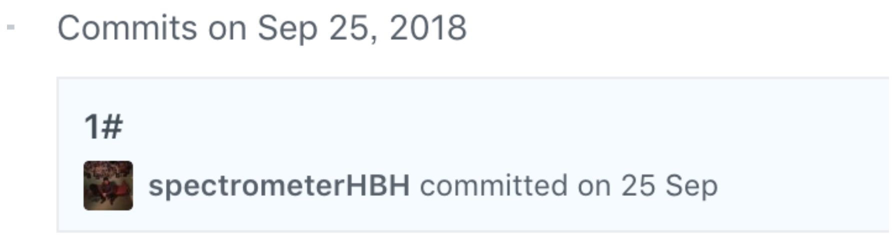
Begining & version1



4-stage pipeline CPU with OoO
By Bohan Hou

- Implement a simplified design at first
- Branch Policy : stall
- ROB only for renaming use and planned to add speculation later on

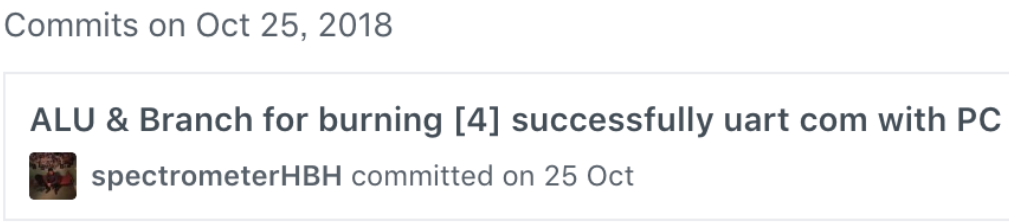
Beginning & version1



Commits on Sep 25, 2018

1#

spectrometerHBH committed on 25 Sep



Commits on Oct 25, 2018

ALU & Branch for burning [4] successfully uart com with PC

spectrometerHBH committed on 25 Oct

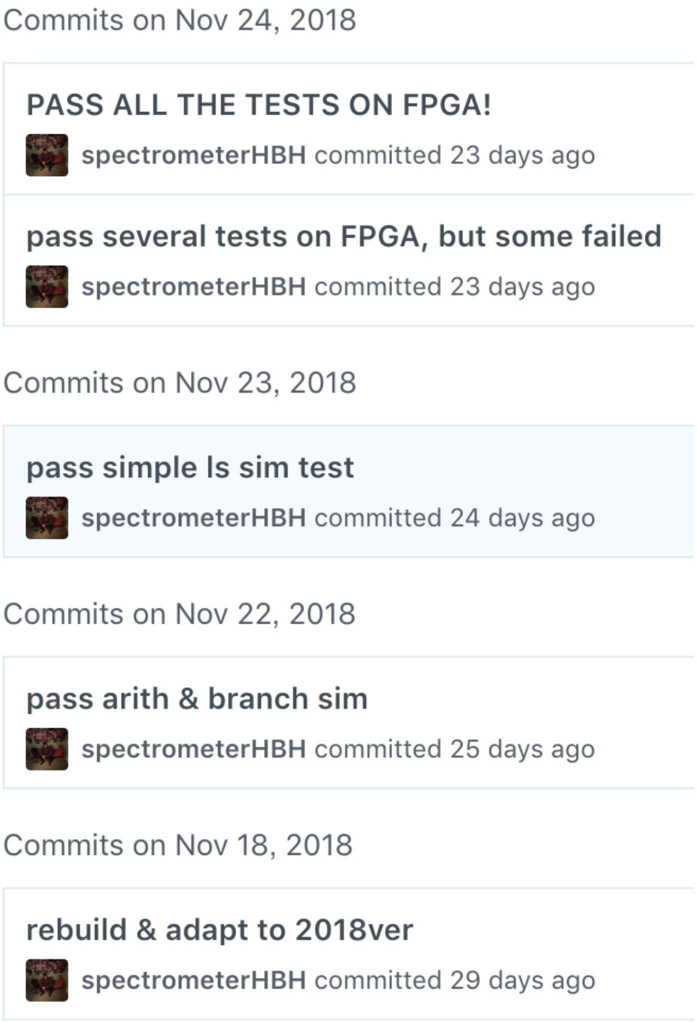
- Sep 25~Oct 25
- arith & branch instructions pass simulation
- System 17 framework (UARTcom with PC, Mem on PC)

Beginning & version1

- Code of poor quality——unfamiliar with VerilogHDL
- "software-style HDL programming"
  HW-style thinking
- Timing design in chaos
  What is a digital system design at all?

  A set of automatons
  ≈A group of people working under command
- TA changed the framework

Rebuild & version2



Commits on Nov 24, 2018

PASS ALL THE TESTS ON FPGA!
spectrometerHBH committed 23 days ago

pass several tests on FPGA, but some failed
spectrometerHBH committed 23 days ago

Commits on Nov 23, 2018

pass simple ls sim test
spectrometerHBH committed 24 days ago

Commits on Nov 22, 2018

pass arith & branch sim
spectrometerHBH committed 25 days ago

Commits on Nov 18, 2018

rebuild & adapt to 2018ver
spectrometerHBH committed 29 days ago

- Nov 18~Nov 24
- Rebuild & load/store instructions complete
- Redesign automatons and timing logic
- Load/Store instructions excecuted in order because of limited memory bandwidth
- Pass all tests on FPGA
- Add a 512B direct indexed I-cache

Rebuild & version2

- What's next?
- Dynamic Branch Prediction?
  - Gshared/Tournament Predictor?
  - Limited LUT sources on FPGA, but not impossible

  - Misprediction Policy?
  - 1.Flush when branch instruction graduate?
  - Hurt performance if misprediction often happens

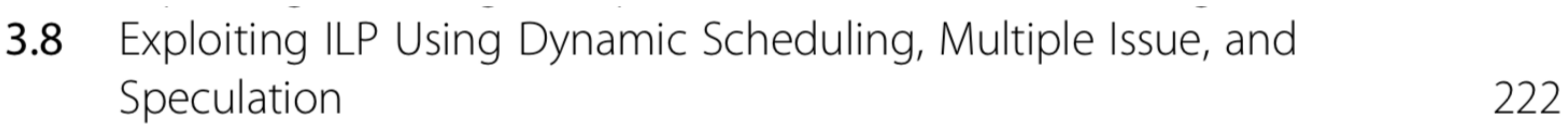
  - 2.Flush when branch target clear?
  - Require rather complicate logic
  - Register & ROB status copy, require large LUT sources
  - Speculation tag for multiple branches on pipeline
  - Integrate ex_ls with ROB
- Makes things rather complicate and not a simple design

Rebuild & version2

- What's next?

- Dynamic Branch Prediction is not an elegant design

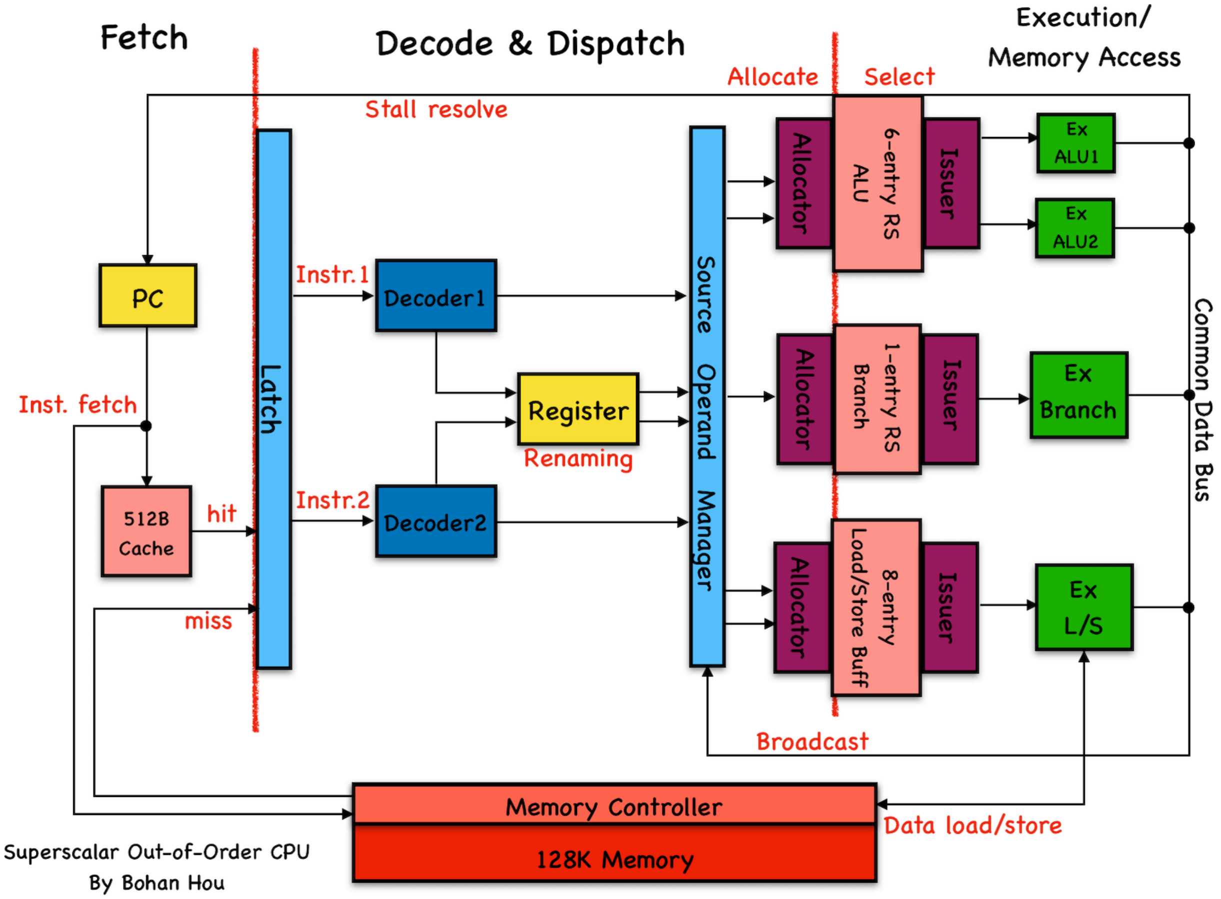
- Decode & Dispatch is bottle neck for CLOCK RATE
  - Instr -> Regfile -> ROB -> Reservation Station



3.8 Exploiting ILP Using Dynamic Scheduling, Multiple Issue, and Speculation 222

- Multiple Issue is attracting

Superscalar & version3

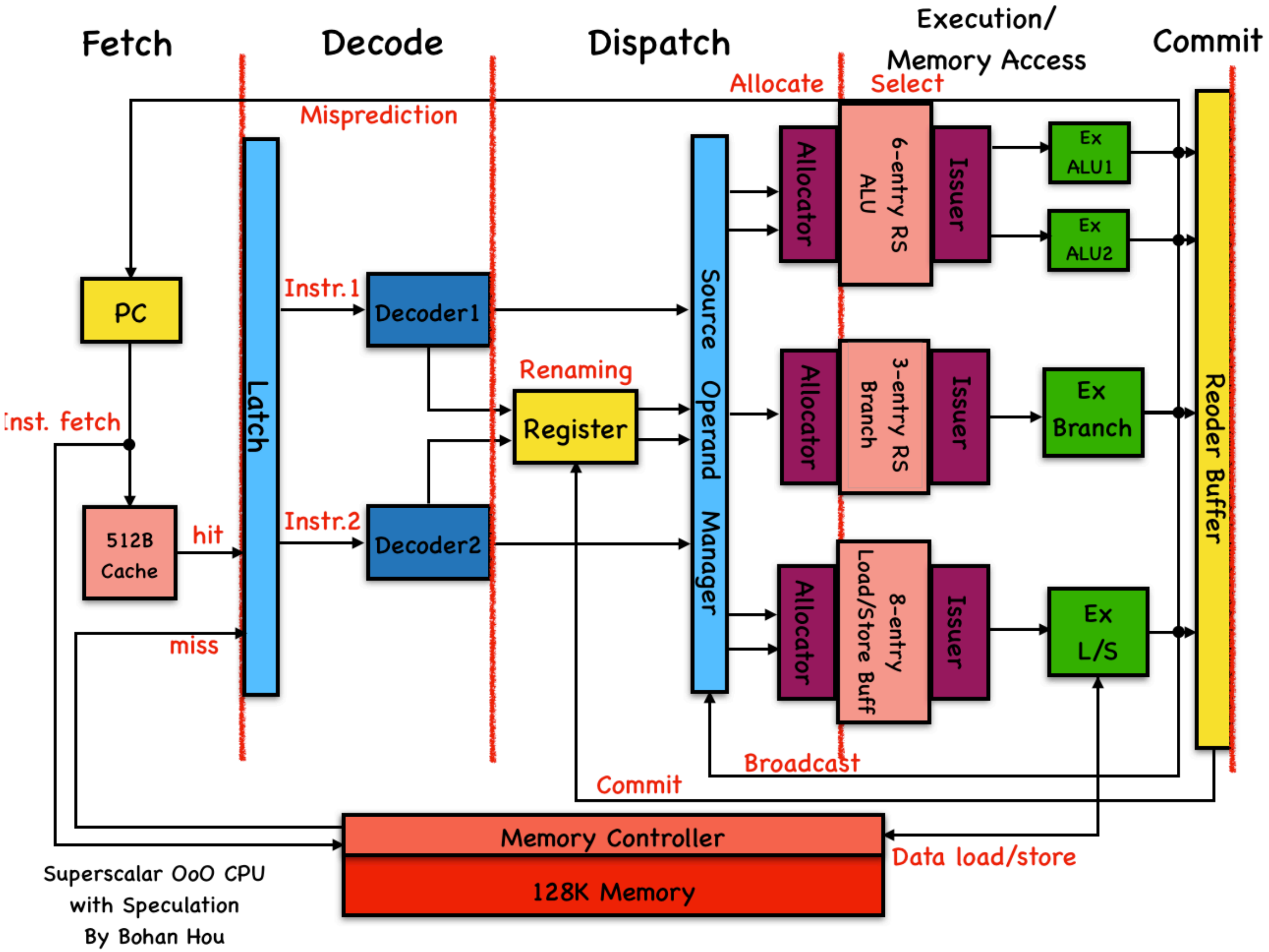


- Dec 4~Dec 10
- Shorten pipeline to 3 stages to decrease branch stall cycles
- Use Reservation Station Number for renaming,
- Remove ROB
- Cache hit makes dual improvement, so I changed DM cache to 2-way SA cache, LRU

Superscalar & version3

- In testcase multiarray.c
  - 60000ns -> 30000ns in simulation
- In testcase pi.c
  - 2.52ns -> 2.15ns -> 2.00ns under 100MHz on FPGA
- Branch stall is still the main problem to tackle
  - Speculation is inevitable
  - Approximately 50% CPU time spends on branch stall
  - LUT 81%

Speculation & version4



- Dec 19~20
- Add ROB
- disjoint Decode & Dispatch
- 6-entry ALU, 3-entry Branch, 8-entry L/S Buf, 32-entry ROB
- Flush when branch commits

Speculation & version4

- It was not a successful try, which was as expected
  - 50MHz, 3.5ns in testcase pi.c
  - My goal is to gain higher performance even under lower CLK Rate
- Branch Accuracy is moderate, but not satisfying
  - gshared(80%~85%) > local 2-bit(70%~75%) > ght(60%~65%)
  - 64-entry on FPGA, but no obviously improval when 4096-entry in simple simulation experiment(not sure)
- Branch Misprediction Penalty is catastrophic
  - L/S is too slow, wrong branch got stuck in ROB

Thinkings on version5……

- Simplify Dispatch Stage to Recover CLK Rate
    - Avoid checking out ROB during Dispatch possible
- Change Misprediction Policy
    - Require many LUT sources
- Load/Store commit faster
    - Another buffer for non-speculative load/store
    - Dcache & write buffer
- No speculation
    - Dcache & simplify logic to improve CLK Rate
    - Shorten branch stalls

- But that's all
    - GPA matters……
    - LUT 92%……

Summary

- Simple often wins
  - Being more complicate has been the tendency till now, at least for processor designs.
  - Humans born greedy.
  - Simplicity is relative

- Performance is the only criterion for designers
  - My design is a failure
  - Slower than standard 5-stage pipeline, which is frustrating

- “People count projects you finish, not the ones you start”
  - Do one thing, do it well

Ending

- That's all for my journey
- Thanks for listening