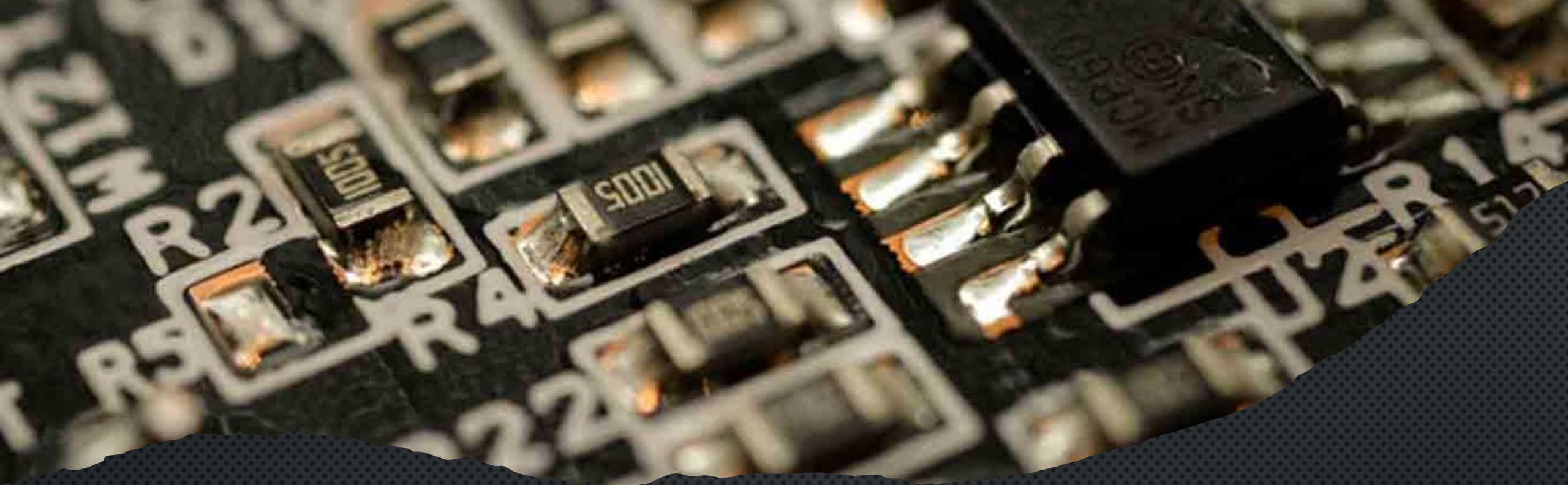




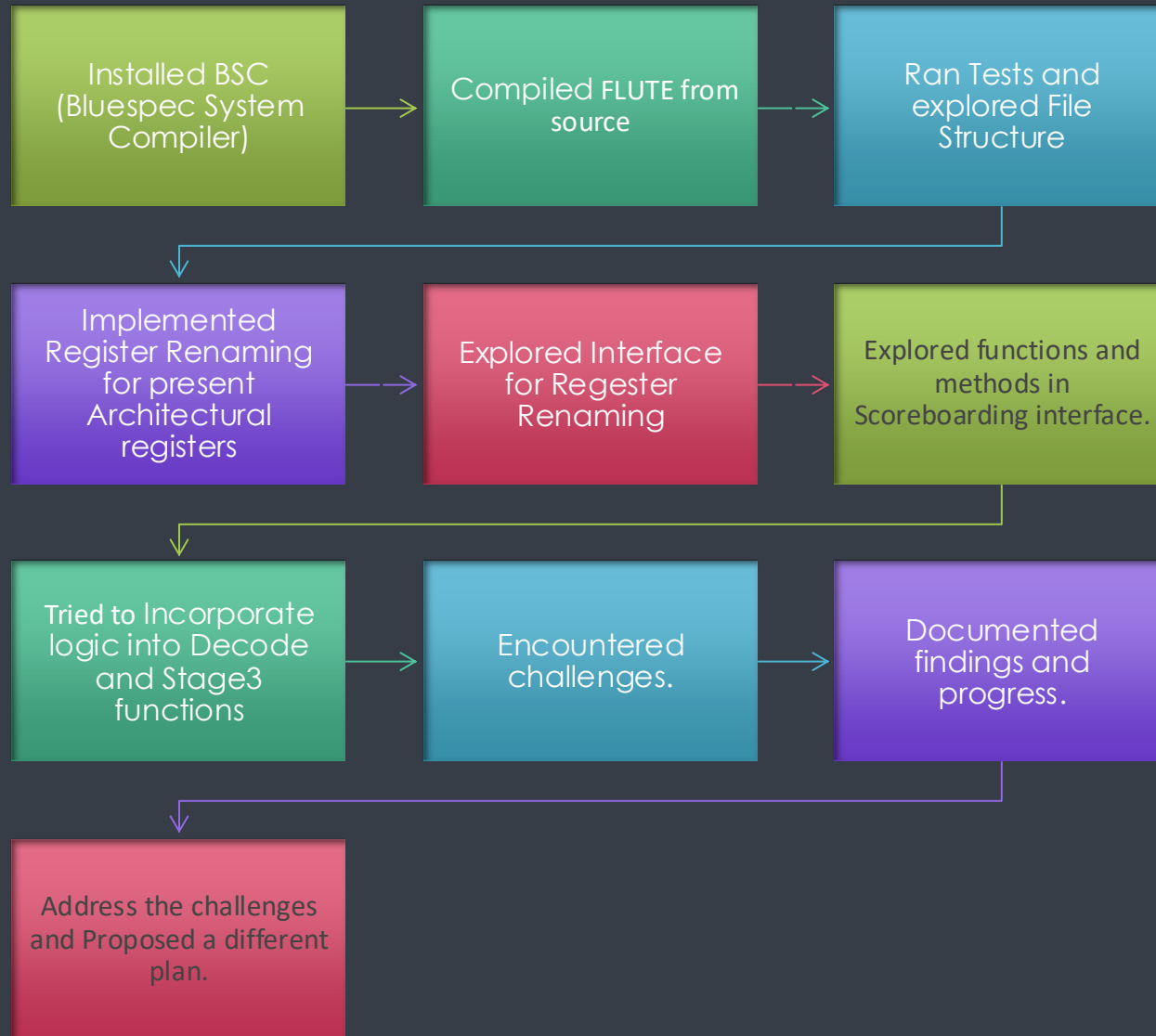
RISC-V PROCESSOR WITH SCOREBOARDING AND REGISTER RENAMING

ARYAN | ABHIRAM



PROJECT DESCRIPTION

- TO IMPLEMENT SCOREBOARDING AND REGISTER RENAMING IN RISC-V CORE.
- TO EVALUATE PERFORMANCE BASED ON METRICS LIKE INSTRUCTION THROUGHPUT, EXECUTION LATENCY.



OVERALL PROGRESS

YOU CAN FIND OUR EFFORTS ON GITHUB...

Aryan40IIIT / enhanced_flute

Search

Type to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

enhanced_flute

Unwatch 1

Fork 0

Star 0

main 1 Branch Tags

Go to file

Add file

<> Code

About

notpua

Execute Stage: change the read registers

984ddcb · 4 minutes ago

7 Commits

Doc/Microarchitecture	initial commit	last week
Tests	initial commit	last week
builds	initial commit	last week
src_Core	Execute Stage: change the read registers	4 minutes ago
src_Testbench	initial commit	3 days ago
src_bsc_lib_RTL	initial commit	3 days ago
.gitattributes	initial commit	5 days ago
.gitignore	Renaming 1	2 hours ago
LICENSE	initial commit	2 years ago
README.adoc	initial commit	2 years ago

README Apache-2.0 license

Flute, a free and open-source RISC-V CPU

Contents

- 1. CPU microarchitecture and options
- 1.1. RISC-V ISA Spec options

No description, website, or topics provided.

Readme

Apache-2.0 license

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

Bluespec 92.6%

Verilog 3.7%

Python 1.5%

C 1.1%

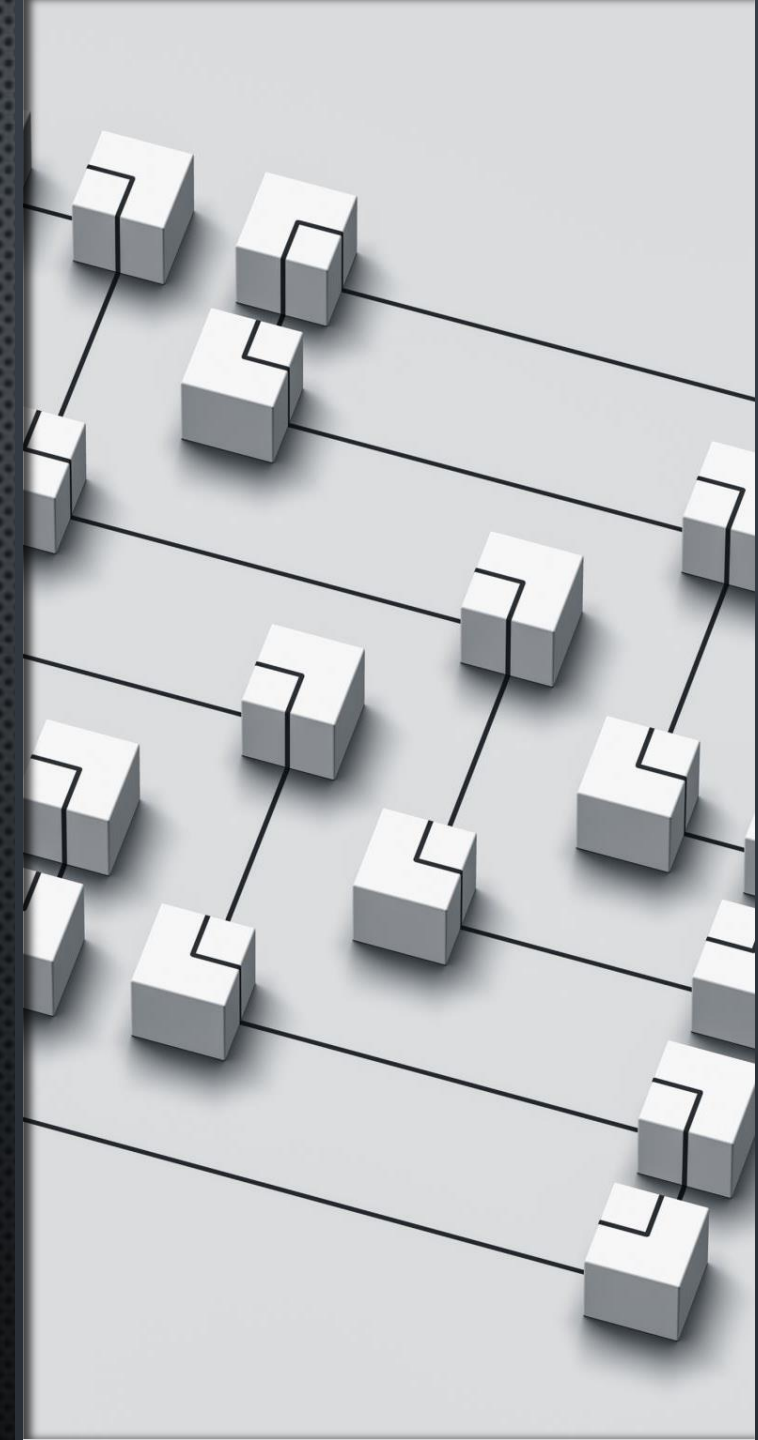
Other 1.1%

Suggested workflows

Based on your tech stack

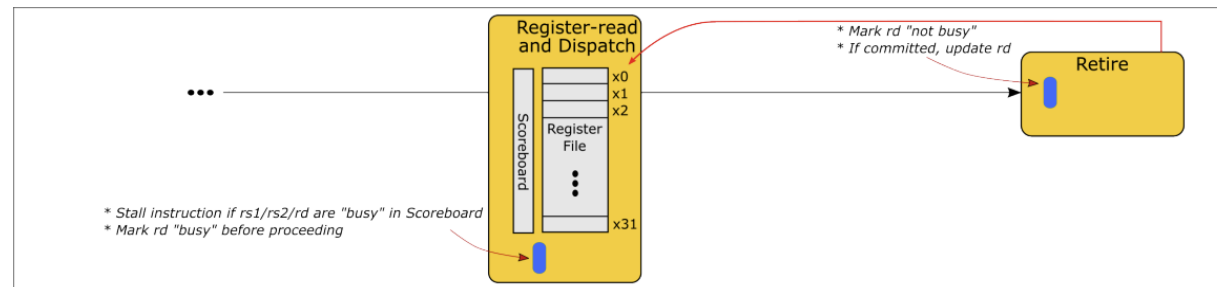
SCOREBOARDING IMPLEMENTATION

- DESIGN:
 - IMPLEMENTED AS AN ARRAY FOR EVERY 1-BIT REGISTER.
 - PREVENTS PREMATURE EXECUTION OF INSTRUCTIONS.
- EXAMPLE:
 - WRITING TO REGISTER x7 SETS SCOREBOARD BIT 7 AS BUSY; SUBSEQUENT READS STALL UNTIL RESOLVED.



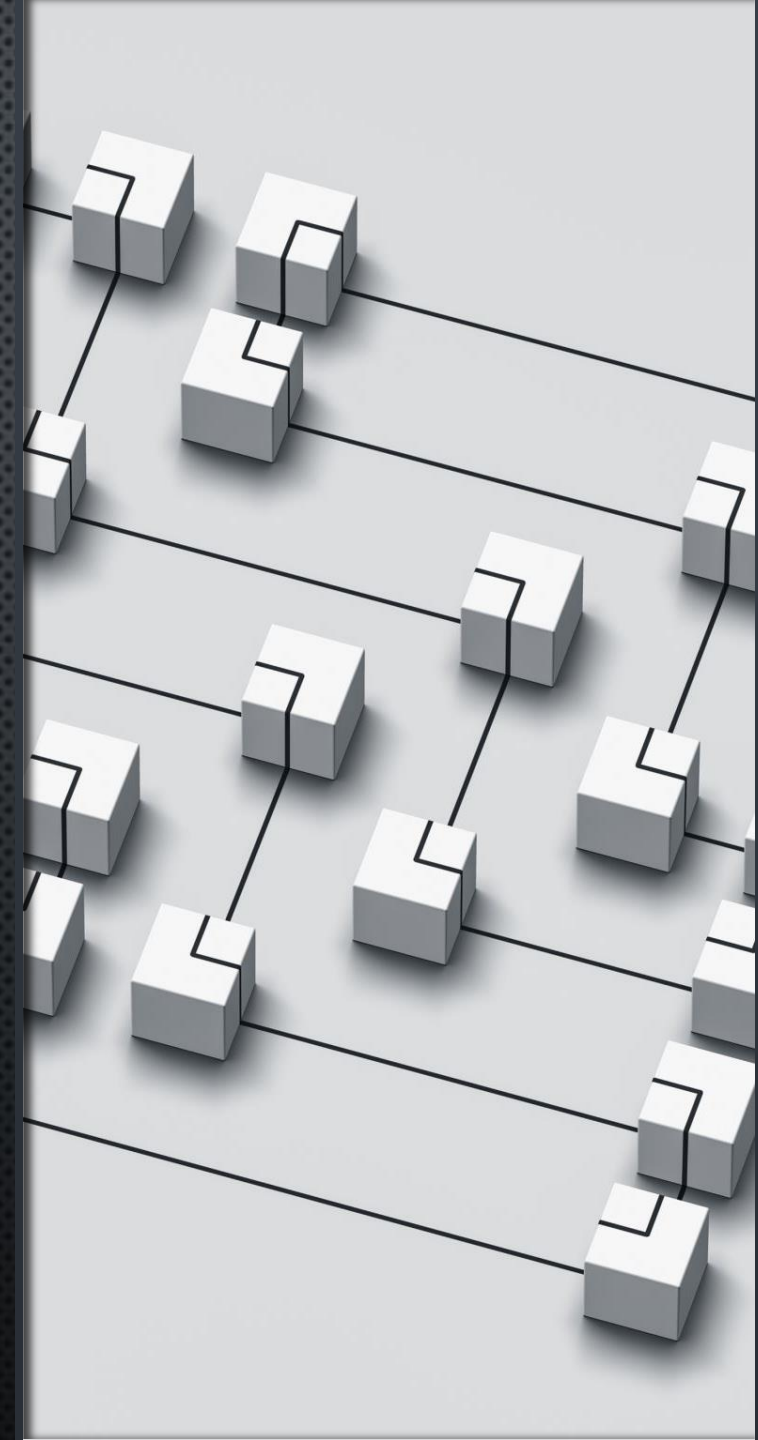
SCOREBOARDING IMPLEMENTATION

- STEPS:
 - INSTRUCTION IN **REGISTER-READ** STAGE SETS THE SCOREBOARD BIT TO 1 (REGISTER MARKED BUSY).
 - AT THE **RETIRE** STAGE, SCOREBOARD BIT RESETS TO 0 (REGISTER FREE).
 - DEPENDENT INSTRUCTIONS STALL UNTIL THE REGISTER IS FREE.
 - FOR ALL THIS WE MADE CHANGES IN DECODE STAGE AND STAGE 3



SCOREBOARDING IMPLEMENTATION EXAMPLE

- **GOAL:**
 - TRANSITION FROM IN-ORDER TO OUT-OF-ORDER EXECUTION.
- **CURRENT STATUS:**
 - ALTHOUGH WE WERE UNABLE TO FULLY TRANSITION TO OUT-OF-ORDER EXECUTION, WE SUCCESSFULLY DEMONSTRATED THE IMPLEMENTATION OF PROPER SCOREBOARDING.
- **CHALLENGES:**
 - REQUIRES ADDITIONAL COMPONENTS LIKE REORDER BUFFERS.
 - SIGNIFICANT MODIFICATIONS NEEDED IN THE EXECUTION AND COMMIT STAGES.



TRIED TO ADD SOME CUSTOM TEST CASES

- **PURPOSE:**
 - TO TEST OUR CODE, WE NEEDED TO TEST THAT UNITS WERE BEING UTILIZED MULTIPLE TIMES.

```
+ #include "riscv_test.h"
+ #include "test_macros.h"
+
+ RVTEST_RV64UF
+ RVTEST_CODE_BEGIN
+
+ TEST_FP_OP2_D( 1, fadd.d, 0,          3.5,      2.5,      1.0 );
+ TEST_FP_OP2_D( 2, fadd.d, 1,        -1234,    -1235.1,    1.1 );
+ TEST_FP_OP2_D( 3, fadd.d, 1,    3.14159266, 3.14159265, 0.00000001 );
+
+ TEST_PASSFAIL
+
+ RVTEST_CODE_END
+
+ .data
+ RVTEST_DATA_BEGIN
+
+ TEST_DATA
+
+ RVTEST_DATA_END
```


LEARNING OUTCOMES



Hardware Knowledge:

Gained insights into CPU architecture and hardware description languages like Bluespec Verilog.



Pipeline Design:

Deepened understanding of pipelining and hazard management.



Techniques:

Learned the intricacies of register renaming and scoreboard mechanisms.



Personal Growth:

Enjoyed the challenge of tackling a real-world hardware project.

THINGS WE GOT STUCK IN

Implementing Out-of-Order Execution:

Despite significant efforts, we were unable to fully transition to out-of-order execution. This challenge stemmed from the complexity of modifying the pipeline to support dynamic instruction scheduling and reordering.



THANK
YOU