Implementation of 32-bit 5-stage pipelined RISC-V processor

by

Manish Kumar Mahato
2018-EE-192

Advisor:
Mr. Umer Shahid

FALL 2021

Department of Electrical Engineering
University of Engineering and Technology, Lahore

Course Code and Title: EE-475: Computer Architecture
Semester: Fall 2021 (7th Semester)
Instructor: Dr. Muhammad Tahir & Mr. Umer Shahid
Total Marks: 50 (in Lab)
Deadline: End of Semester

CLOs and PLOs for Complex Engineering Problem

Below stated are the CLOs and PLOs addressed in the complex engineering problem along with domain and level. These are the CLOs from the theory/lab course which are already defined.

CLOs		**Description**	**Domains & Levels**	**PLOs, Levels**
CLO1	Theory	Demonstrate the skills to design Datapath of single-cycle, multi-cycle, and pipeline microprocessor architecture using a variety of techniques	Cognitive, 2. Understand	PLO1
CLO2	Lab	Formulate the control portion of a processor, and its data path consisting of the general-purpose register file, ALU, the memory interface, internal registers between stages, and multiplexers.	Psychomotor, 4. Articulate	PLO3
CLO3	Lab	Design and emulate a pipelined CPU by given specifications using Verilog	Psychomotor, 5. Naturalization	PLO3

Problem Statement

In this open-ended design lab, students will investigate and design a 32-bit RISC-V processor with five stage pipeline.

- Implement a processor that supports a subset of the RISC-V ISA. The designed processor does not support all RISC-V ISA instruction, however it will have most parts that a real processor has: A fetch unit, decode logic, functional units, a register file, I/O support, access to memory, interrupt handling, hazard elimination protocols, and CSR Support.

- Students will be implementing the Datapath while designing a pipelined RISC-V processor which will work with five stages mainly, it must contain a file or block RAM of FPGA that will be used as Random-Access Memory. It will have 32 Registers working as General-purpose Register. While some special purpose registers (Program Counter, CSR registers) will be used to hold the address of the instruction and will handle the interrupts. Each Register must be 32 bit wide and clock edge triggered.

ABSTRACT

RISC-V processor is implemented. It has five stages pipelined. It supports r-type, i-type, s-type, b-type and j-type instructions of RISC-V. It has hazard unit and CSR unit. The working of code has been verified using Cadence software.

ACKNOWLEDGMENTS

I would like to thank Dr. Muhammad Tahir and Mr. Umer Shahid for teaching us the course and guiding us throughout the course. I would also like to appreciate my friends Ali Tariq and Ali Imran who helped me resolve my problems when it was difficult for me to understand.

STATEMENT OF ORGINALITY

It is stated that this CEP presented in this course consists of my own ideas and hands-on working. The contributions and ideas from others have been duly acknowledged and cited in the dissertation. This complete report and code is written by me.

[Manish Kumar Mahato]

TABLE OF CONTENTS

1. INTRODUCTION

RISC-V processor is implemented in hardware descriptive language Verilog. Vivado was used for implementing the processor. First a Datapath was made which consist of various parts of processor such as Instruction Memory, Data Memory, ALU (Arithmetic and Logic Unit), Register File, Mux. Then Control Unit was made according to the instructions we wanted to use. Single Cycle Processor was developed first which was later converted into Pipelined Processor. Then Branch Prediction was added to the design. Hazard Unit and CSR module was configured at the end. The processor is based on Harvard Architecture.

1.1. Design Description

This processor can be operated on different kinds of RISC-V instructions such as r-type, i-type, s-type, b-type and j-type. Data and control hazards are handled by hazard controller while interrupts are handled by CSR module.

1.2. Potential Hazards

In this processor there is no any occurrence of structural hazards but there is data and control hazards. These potential hazards are handled by hazard unit. The reason for no structure hazard is use of separate data memory and instruction memory.

1.3. Datapath Explanation

This Datapath consists of various modules which are connected together for better working. The separate modules are in different phases. We have memories: instruction memory and data memory. We have ALU which does all kind of arithmetic and logic operations. We have register file and immediate generator. We have control unit, hazard unit and CSR unit.

1.3.1. Highlighted Datapath for R-type Instructions

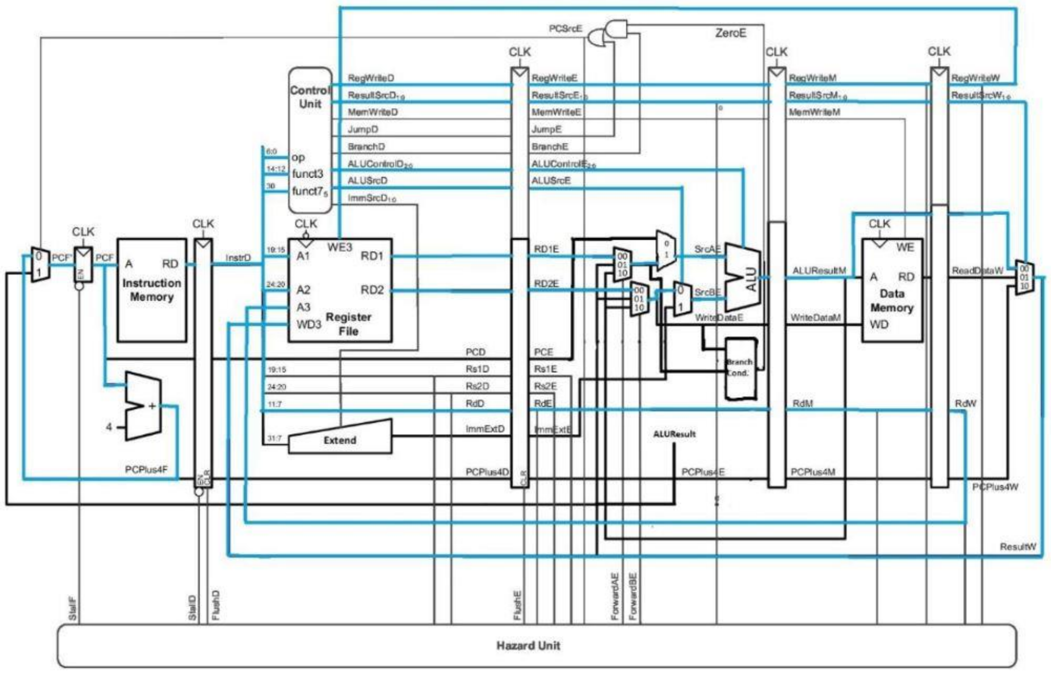


Figure 1.1 Datapath for r-type instructions

1.3.2. Highlighted Datapath for I-type Instructions

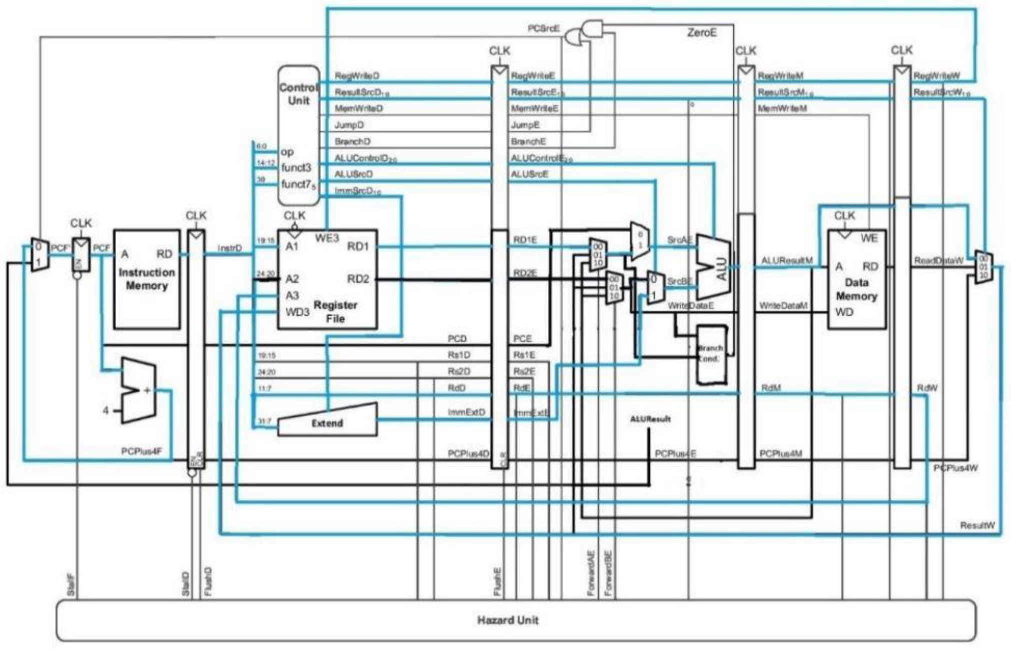


Figure 1.2 Datapath for i-type instructions

1.3.3. Highlighted Datapath for S-type Instructions

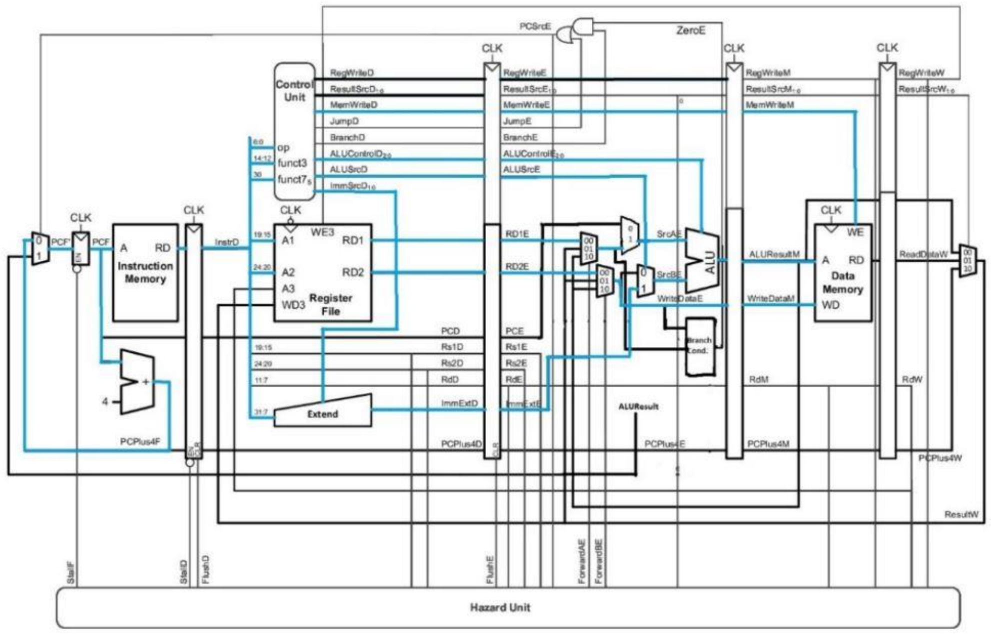


Figure 1.3 Datapath for s-type instructions

1.3.4. Highlighted Datapath for B-type Instructions

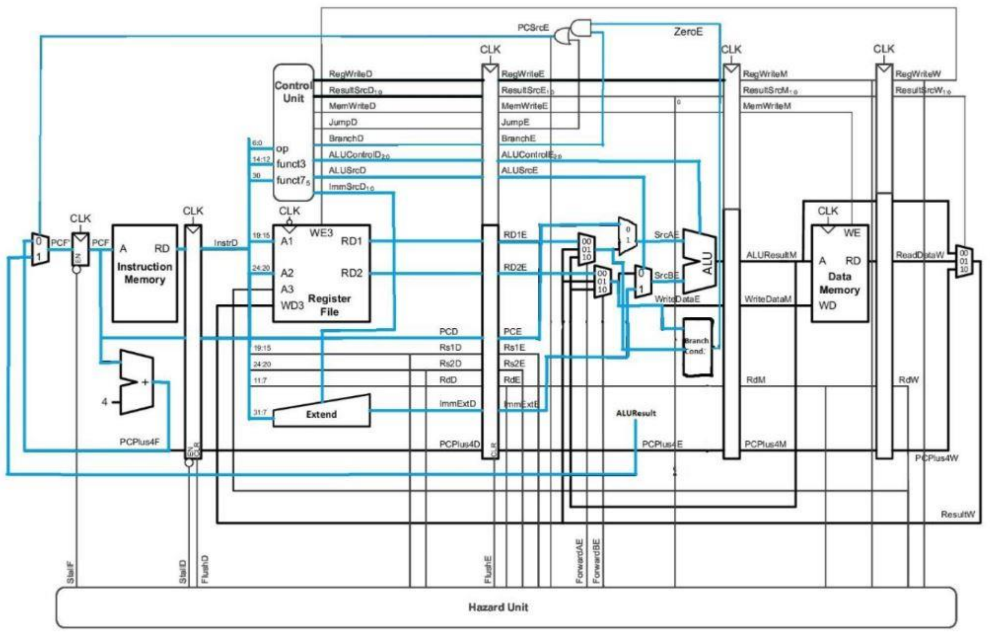


Figure 1.4 Datapath for b-type instructions

1.3.5. Highlighted Datapath for J-type Instructions

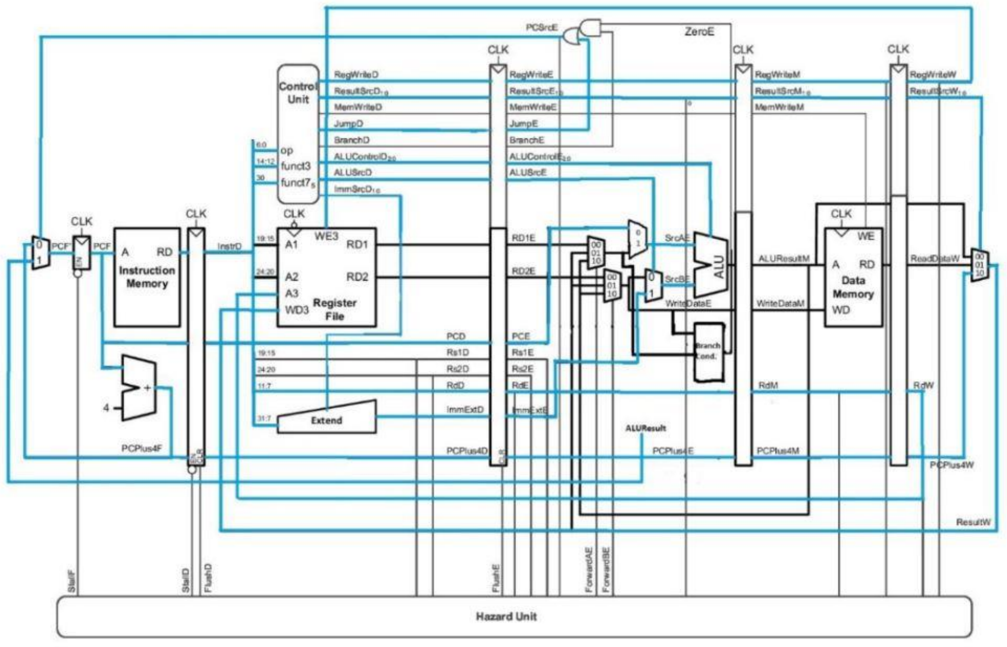


Figure 1.5 Datapath for j-type instructions

1.3.6. Highlighted Datapath for Hazards

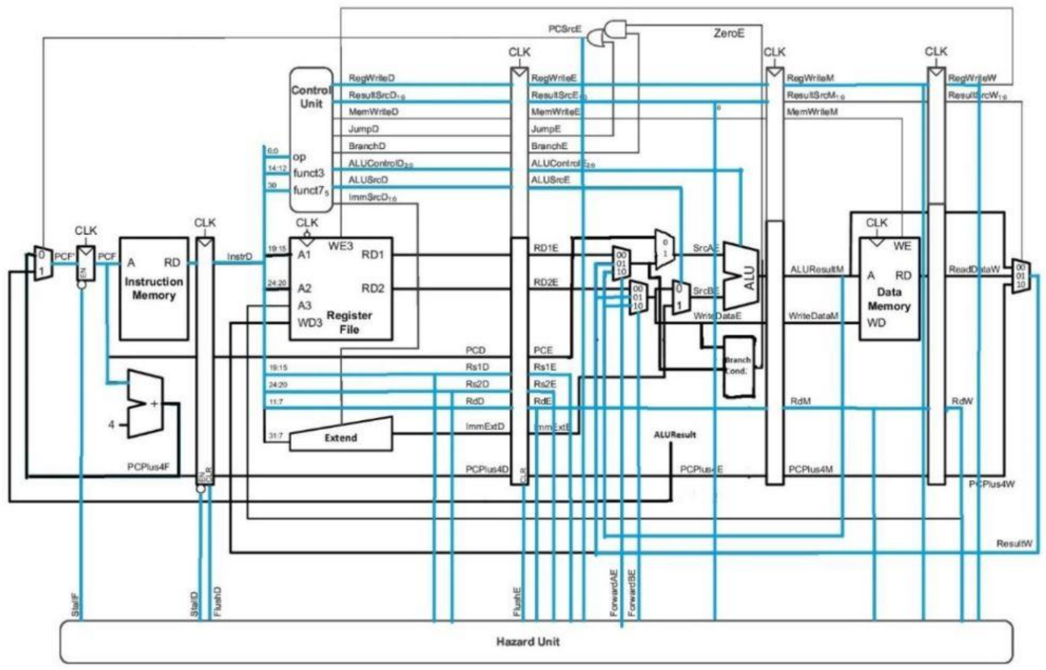


Figure 1.6 Datapath for hazards

2. Modules

The description of each modules is written briefly below:

2.1. The Top module

The name of the Top module in this Project is given BasicStructure.v. It consists of code of seven segment which is configured to the FPGA board for implementation check on FPGA. Then it consists of the connection of each modules by calling them. They are joined by using dot convention. Control Unit code is also placed inside this module. I have also placed mux code in this module at the end.
After code of control unit there is also part for b-type instruction and part of branch prediction.

2.2. CPU Control Unit

Control unit is placed inside the top module BasicStructure.v, I have not made separate module for controller alone. Input to the control unit are opcode, func3 and func7. These are used to find what kind of instruction is being fetched. After this the controller will give the output signals to the various modules accordingly which should be being used for specific kind of instructions based on input opcode, func3 and func7.

2.3. Memory Module

The memory module is supporting both loads and stores. The read in memory is asynchronous while write is synchronous.

2.4. Register File

There are total 32 registers in the register file. These registers are also of size 32bits. There are 5-bit address ports, 2 32bit read address ports, and one 32bit write address port. Register file has two output ports.

2.5. ALU

The ALU is of 32 bits and it performs various arithmetic and logic operations such as add, subtract, multiply, divide, or, and, nor, not etc. It has 2 inputs and one output.

2.6. Hazard Controller

The hazard unit is made as a hazard controller, it solves data hazards and control hazards.

2.7. Forwarding Unit

Forwarding is used for forwarding the data or information required in next pipelined stage as soon as data arrives at it execute stage.

2.8. CSR Controller

This module is implemented which takes interrupt and exceptions as its input and provides the output to the mapped modules which takes care of exception and interrupt which has occurred.

2.9. CSR Control Signals

A separate module is made for CSR named as CSR module in code. It makes the interrupt signal which is sent to hazard unit. It has interrupt, clk and mret as its input.

3. Tests

The processor was tested on Vivado and cadence for its proper working and verification.

3.1. Simulation Tool Description

Simulation tool which was used was Xilinx Vivado.

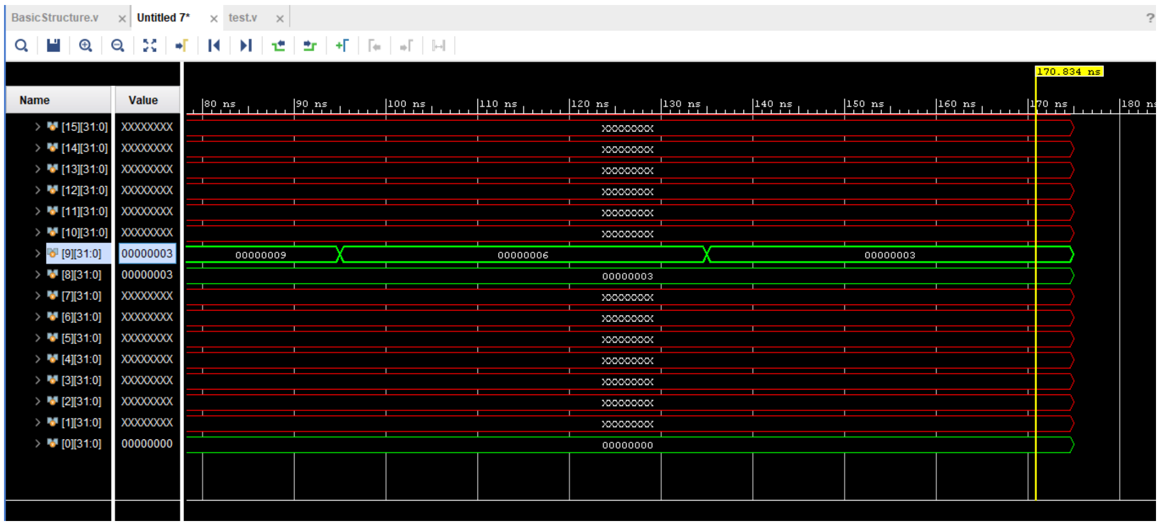
3.2. Steps to run a basic modular test

a. The code of GCD was converted to machine code and placed in instruction memory which had hazards in it.

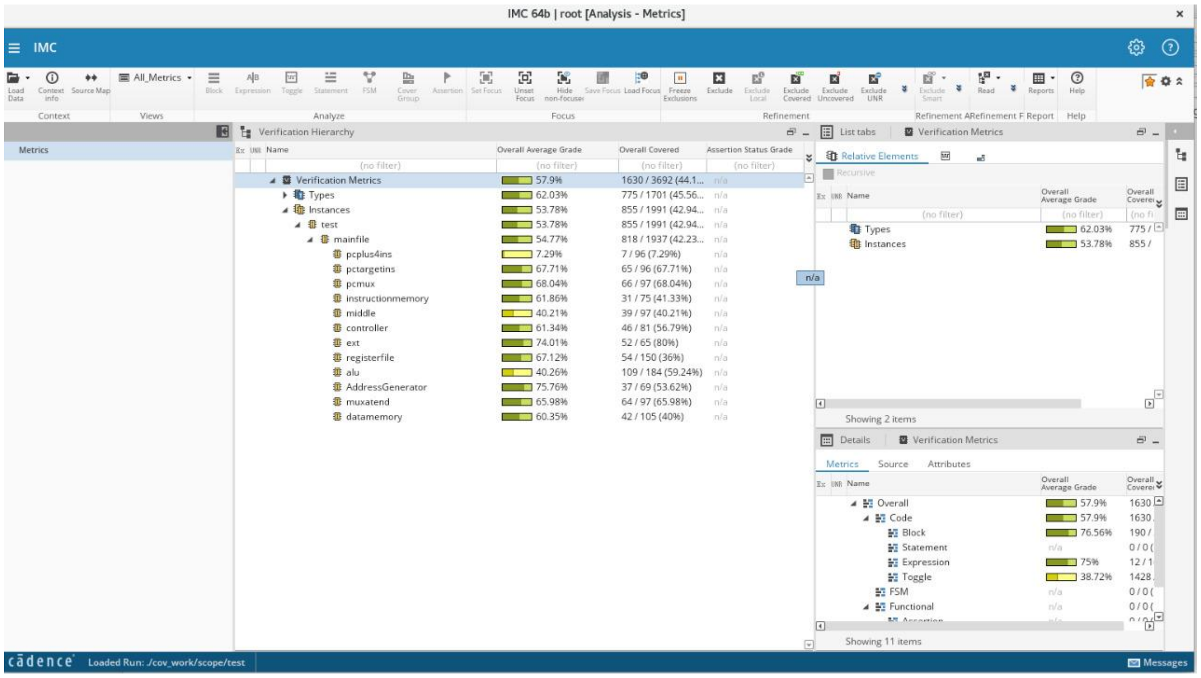
b. Simulation was done on Vivado and checked on testbench if it was giving proper results.

3.3. Test performed

Output for GCD of the testbench which is written in test.v was seen as:



3.4. Coverage Report



Loaded Run: ./cov_work/scope/test

Name	Overall Average Grade	Overall Covered	Assertion Status Grade
Verification Metrics	57.9%	1630 / 3692 (44.1...	n/a
Types	62.03%	775 / 1701 (45.56...	n/a
Instances	53.78%	855 / 1991 (42.94...	n/a
test	53.78%	855 / 1991 (42.94...	n/a
mainfile	54.77%	818 / 1937 (42.23...	n/a
pcplus4ins	7.29%	7 / 96 (7.29%)	n/a
pctargetins	67.71%	65 / 96 (67.71%)	n/a
pcmux	68.04%	66 / 97 (68.04%)	n/a
instructionmemory	61.86%	31 / 75 (41.33%)	n/a
middle	40.21%	39 / 97 (40.21%)	n/a
controller	61.34%	46 / 81 (56.79%)	n/a
ext	74.01%	52 / 65 (80%)	n/a
registerfile	67.12%	54 / 150 (36%)	n/a
alu	40.26%	109 / 184 (59.24%)	n/a
AddressGenerator	75.76%	37 / 69 (53.62%)	n/a
muxatend	65.98%	64 / 97 (65.98%)	n/a
datamemory	60.35%	42 / 105 (40%)	n/a

Name	Overall Average Grade	Overall Covered
Types	62.03%	775 /
Instances	53.78%	855 /

Showing 2 items

Name	Overall Average Grade	Overall Covered
Overall	57.9%	1630
Code	57.9%	1630
Block	76.56%	190 /
Statement	n/a	0 / 0 (
Expression	75%	12 / 1
Toggle	38.72%	1428
FSM	n/a	0 / 0 (
Functional	n/a	0 / 0 (

Showing 11 items

3.5. Challenges and Limitations

Challenges and limitation are written below:

3.5.1. Challenges

Challenge was finding where the actually output went wrong. After writing the modules which seemed totally correct was not giving proper outputs which took a lot of time to debug.

3.5.2. Limitations

Many of the RISC-V instructions are covered but still all of the RISC-V instructions were not included here.

BIBLIOGRAPHY

[1] S. Harris and D. Harris, Digital Design and Computer Architecture RISC-V Edition, 2006.

VITA

I am Manish Kumar Mahato, I am a final year student of BS Electrical Engineering at UET, Lahore. I am from Nepal and I am an international student here at UET. I passed my High School in Hetauda School of Management and Social Sciences, Nepal in 2017. I got scholarship for UET from Ministry of Education in Nepal. I will finish my BS in Electrical Engineering in 2022.