Experiment No 7

Design of Single Cycle Processor Data path I

7.1 Objectives:

After completing this experiment, student will be able to:

- Understand the basic functions of Data Path.
- Design a Register File in LogiSim.
- Design a complete ALU in LogiSim.

7.2 Background Theory

A data path is the collection of hardware components within a CPU that performs the actual data processing operations, including arithmetic, logic, memory access, and data transfer. To design a data path for executing RISC-V instructions, begin by identifying the major components required for each instruction type and their control signals, starting with the instruction memory, program counter (PC), and an adder to increment the PC. The memory unit stores program instructions, and the PC holds the address of the current instruction, incremented by the adder for the next instruction. These components form the initial part of the data path for fetching instructions. For R-format instructions, which perform arithmetic or logical operations on two registers and write the result to a third, we need a register file and an ALU. The register file, which stores the processor's 32 general-purpose registers, allows reading and writing by specifying register numbers. The ALU performs operations like addition or subtraction on the values read from the registers. Each R-format instruction requires reading two data words from the register file and writing one data word back, managed by specific control signals. The register file's inputs specify the registers to read and write, while the outputs carry the data. The ALU takes two 32-bit inputs, produces a 32-bit result, and outputs a signal if the result is zero, controlled by a 4-bit signal. Integrating these components ensures the data path can fetch, decode, and execute instructions correctly.

7.2.1 Designing of a Register File

Designing a register file in Logisim for a 16-bit single-cycle RISC-V processor involves creating a set of registers that can be read from and written to, controlled by the appropriate control signals.

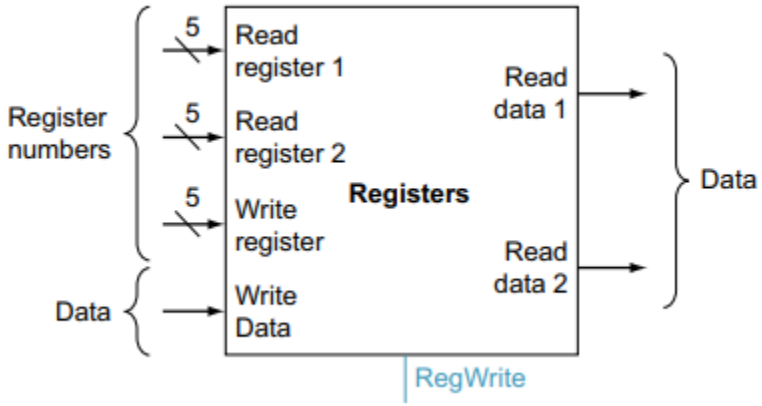


Figure 7.1: 16-Bit Register File Block Diagram

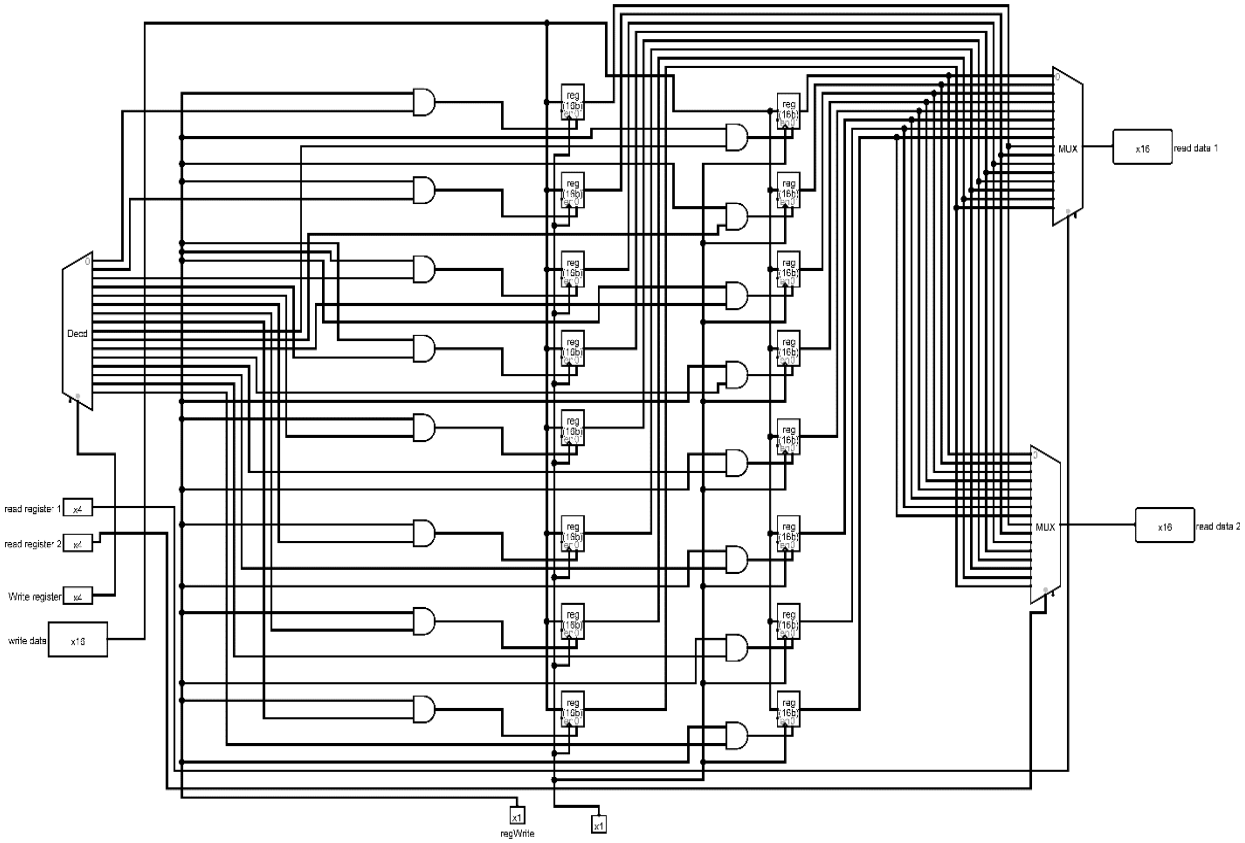


Figure 7.2: 16-Bit Register File Design in LogiSim

7.2.1.1 Step-by-Step Guide to Designing a Register File in Logisim.

1. **Open Logisim:**

- Launch Logisim and create a new project.

2. **Set Up the Circuit:**

- Create a new circuit within your project and name it `RegisterFile`.

3. **Add Registers:**

- Use the "Memory" component library.

- Select the "Register" component.

- Place 16 registers on the canvas to represent the 16-bit registers required for the RISC-V processor.

4. **Configure Registers:**

- Each register should be 16 bits wide. Double-click on each register to set its data width to 16 bits.

5. **Arrange the Registers:**

- Arrange the 16 registers in a column or row to make the design organized and clear.

6. **Create Input and Output Buses:**

- Use "Splitter" components to create input and output buses for addressing and data lines.

- Splitters will help manage the 16-bit data inputs and outputs and the 4-bit register address lines (since 2^4 = 16).

7. **Address Inputs:**

- Add two input pins for the read addresses (`ReadAddr1` and `ReadAddr2`) and one input pin for the write address (`WriteAddr`).

- These address inputs should be 4 bits wide each.

8. **Data Inputs and Outputs:**

- Add one input pin for the write data (`WriteData`), 16 bits wide.

- Add two output pins for the read data (`ReadData1` and `ReadData2`), each 16 bits wide.

9. **Control Signals:**

- Add an input pin for the write enable signal (`WriteEnable`), 1 bit wide.

10. **Connecting Components:**

- Connect the address input pins (`ReadAddr1`, `ReadAddr2`, and `WriteAddr`) to the splitters, and from the splitters to the address inputs of the respective registers.

- Connect the `WriteData` input pin to another splitter, and from there to the data inputs of all registers.

- Connect the `WriteEnable` input to each register's enable pin.

11. **Multiplexers for Read Outputs:**

- Use two 16-bit multiplexers (MUX) to select the output from the appropriate register based on the read addresses.

- Connect the outputs of all 16 registers to the inputs of the multiplexers.

- Connect the `ReadAddr1` and `ReadAddr2` inputs to the selection lines of the multiplexers.

- Connect the outputs of the multiplexers to the `ReadData1` and `ReadData2` output pins.

12. **Final Adjustments:**

- Ensure all connections are correct and labels are clear for easy understanding.

- Test the register file by creating a simple test circuit that writes and reads values from the registers.

7.2.2.2 Example Visual Layout:

- **Address Inputs:** Three input pins (`ReadAddr1`, `ReadAddr2`, `WriteAddr`), each connected to 4-bit splitters.

- **Write Data Input:** One 16-bit input pin (`WriteData`), connected to a 16-bit splitter.

- **Write Enable:** One 1-bit input pin (`WriteEnable`), connected to the enable pins of all registers.

- **Registers:** 16 registers, each 16 bits wide, connected to the address and data splitters.

- **Read Outputs:** Two 16-bit multiplexers, each selecting one of the 16 registers based on the read address inputs, and connected to `ReadData1` and `ReadData2` output pins.

By following these steps, you will create a functional register file in Logisim for a 16-bit single-cycle RISC-V processor, capable of reading and writing data as required by the processor's operations.

7.2.3 16 Bit ALU Design

Designing a 16-bit Arithmetic Logic Unit (ALU) with basic operations like AND, OR, XOR, addition, subtraction, and multiplication involves careful arrangement of logic components. In the following Figure: 2.2, two multiplexers (MUX) play a crucial role in managing the output and carry-out signals. For each operation, the ALU utilizes logic gates such as AND, OR, and XOR to process the input bits. The addition and subtraction operations employ a 16-bit full-adder for the arithmetic computations. To facilitate the selection of the desired operation, a control line is connected to the input of the MUX responsible for generating the output. Another MUX is incorporated to manage the carry-out signal. During addition or subtraction, the carry-out from the full-adder is selected, while for logical operations, the carry-out is set to zero. By implementing this design, the 16-bit ALU effectively performs various operations, and the multiplexers contribute to the flexibility of choosing the appropriate results based on the operation selected.

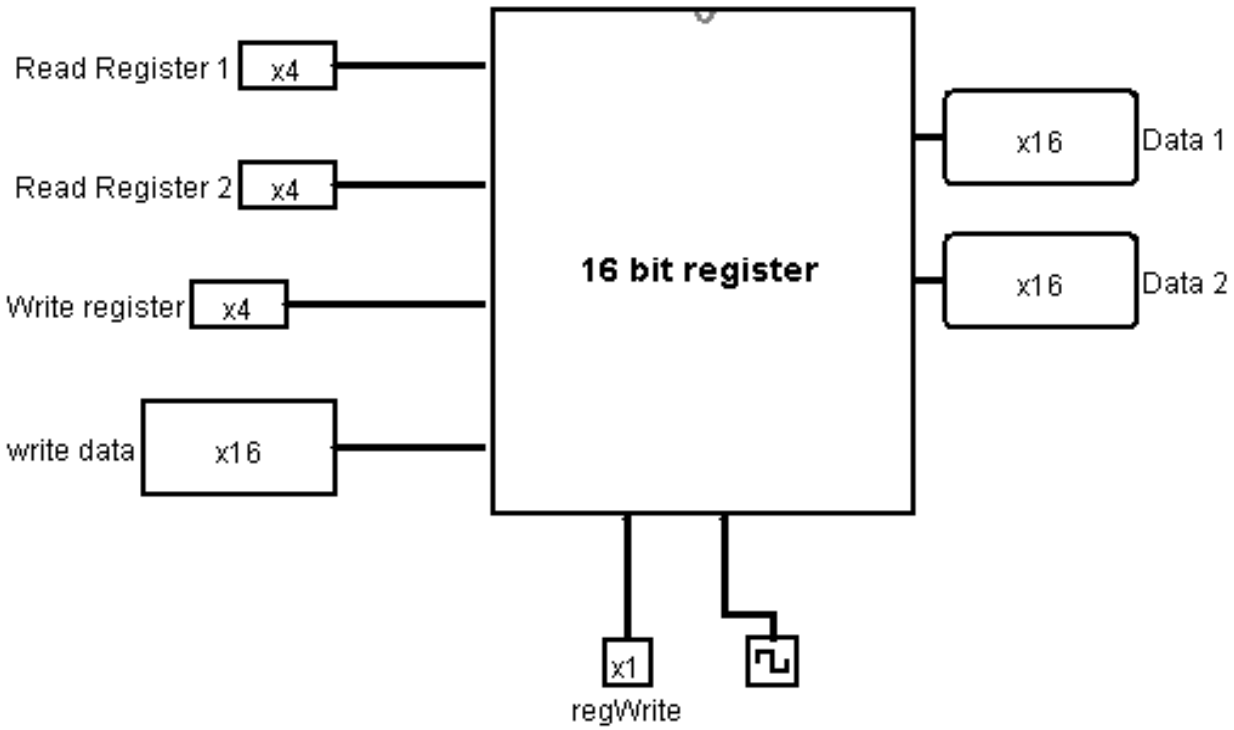


Figure 7.3: 16-Bit ALU Design in LogiSim

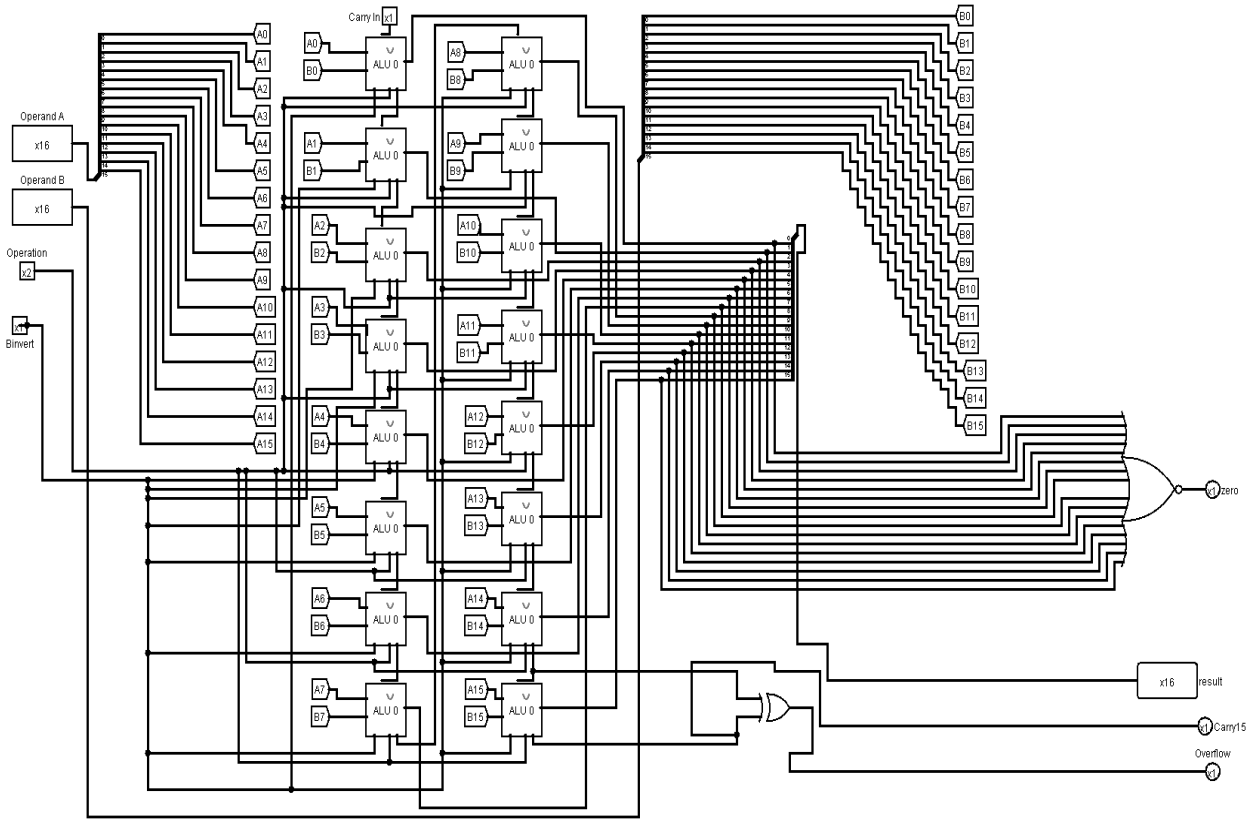


Figure 7.4: 16-Bit ALU Internal Design

7.3 Lab Activities

7.3.1 Task-1: Design and Implement a complete 16-Bit ALU in LogiSim.

7.3.2 Task-2: Design and Implement 16-Bit Register File in LogiSim.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature: _____ **Date: _____**

Cognitive Activity

Q1. For the instruction 'add x5, x6, x7' passing through your data path, trace step by step: which register-file read ports are used, what values come out of them, what the ALU does with those values, and which port is used to write back the result. **(5)**

Ans: _____

_____

_____

_____

_____

Q2. Explain why the register file needs two read ports but only one write port to correctly execute R-type instructions in a single-cycle RISC-V processor. **(5)**

Ans: _____

_____

_____

_____

_____

Obtained Marks	**/10**

Psychomotor Activity 1 — Experimental Setup

S. No	Criteria	Performance	Description	Marks
1	Tool Setup (4)	Level 4	Opens and configures LogiSim and selects the required environment correctly and independently.	4
		Level 3	Configures the required tool with minor errors or assistance.	3
		Level 2	Completes the setup with considerable assistance.	2
		Level 1	Setup is incomplete and requires substantial assistance.	1
2	File/Design Setup (4)	Level 4	Correctly creates/selects the required assembly file, Logisim circuit/design and required components for the Logisim data path design.	4
		Level 3	File/design setup is mostly correct with minor errors or omissions.	3
		Level 2	Setup is partially completed and requires assistance.	2
		Level 1	Major errors or omissions are present in the setup.	1
3	Parameter Setup (2)	Level 2	Correctly defines required data, registers, parameters, components and initial conditions.	2
		Level 1	Most required inputs/components are correctly configured with minor errors.	1
			Total Obtained	/10

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	0

Psychomotor Activity 2 — Implementation + Results

S. No	Criteria	Performance	Description	Marks
1	**Implementation (4)**	Level 4	Implements the required Logisim data path design correctly and independently.	**4**
		Level 3	Implements the required program/design with minor errors or assistance.	**3**
		Level 2	Partially implements the program/design and requires considerable assistance.	**2**
		Level 1	Implementation is largely incomplete and requires substantial assistance.	**1**
2	**Execution/Simulation (2)**	Level 2	Executes the RISC-V program or simulates the Logisim design correctly and handles basic errors appropriately.	**2**
		Level 1	Executes/simulates the work with minor errors or assistance.	**1**
3	**Results/Output (4)**	Level 4	Obtains correct results and clearly demonstrates the required program output or circuit behavior.	**4**
		Level 3	Obtains mostly correct results with minor errors.	**3**
		Level 2	Obtains partially correct results with noticeable errors or omissions.	**2**
		Level 1	Results are largely incorrect or incomplete.	**1**
			Total Obtained	**/10**

If a student is absent:

Criteria	Description	Marks
Absent	Student is absent from the laboratory activity.	**0**

Affective Activity — Lab Report + Viva Voce

S. No	Criteria	Performance	Description	Marks
1	**Lab Report Completeness & Organization (4)**	Level 4	Report is complete, well organized and demonstrates full responsibility for the assigned work.	**4**
		Level 3	Report is complete with minor omissions or organizational issues and demonstrates satisfactory responsibility for the assigned work.	**3**
		Level 2	Report is partially complete with several omissions and demonstrates limited responsibility for the assigned work.	**2**
		Level 1	Report has major omissions or poor organization and demonstrates minimal responsibility for the assigned work.	**1**
2	**Code/Design, Results & Conclusion (2)**	Level 2	Logisim data path design, results, and conclusion are correctly presented and discussed, demonstrating ownership of the assigned work.	**2**
		Level 1	Logisim data path design, results, or conclusion are presented with minor errors or omissions.	**1**
3	**Viva Voce (4)**	Level 4	Clearly explains the assigned Logisim data path design, working, and results, demonstrating individual responsibility, participation, and effective communication.	**4**
		Level 3	Explains most of the assigned program/design and results correctly, with minor difficulty, and demonstrates good individual participation.	**3**
		Level 2	Explains basic aspects of the assigned program/design with some prompting and demonstrates basic participation.	**2**
		Level 1	Demonstrates limited understanding or involvement and requires substantial prompting to explain the assigned work.	**1**
			Total Obtained	**/10**

The student has not submitted the report and has not appeared in the Viva Voce.

Criteria	Description	Marks
Lab Report and Results Not Submitted	Student has not submitted the report and results file.	**0**
Absent in Viva Voce	Student is absent from the viva voce.	**0**

Instructor's Signature: _____ Date: _____