PIPELINED PROCESSOR DESIGN

🔹 Overview

This project focuses on the **design and simulation of a 4-stage pipelined processor** using **Verilog HDL**.
The processor supports basic instructions such as **ADD, SUB, and LOAD**.
Pipelining improves performance by allowing multiple instructions to be executed concurrently in different stages.

🔹 Problem Statement

Modern processors require high performance and efficient instruction execution. The challenge is to design a **simple pipelined processor** that demonstrates instruction flow across pipeline stages while maintaining correct execution and register updates.

🔹 Objectives

- Design a **4-stage pipelined processor** using Verilog HDL
- Implement basic instructions: **ADD, SUB, LOAD**
- Develop a **testbench** to verify functionality
- Observe and analyze **simulation waveforms**
- Understand **pipeline registers and instruction flow**

🔹 Tools Used

- **Verilog HDL** – RTL design
- **Xilinx Vivado** – Simulation and waveform analysis
- **Vivado Simulator** – Behavioral simulation

🔹 Processor Architecture

The pipelined processor consists of the following stages:

1. **Instruction Fetch (IF)**
    - Fetches instruction from instruction memory
    - Uses Program Counter (PC)

2. **Instruction Decode (ID)**
    - Decodes opcode and register fields
    - Passes instruction to execute stage
3. **Execute (EX)**
    - Performs arithmetic or memory access
    - ADD, SUB via ALU
    - LOAD reads from data memory
4. **Write Back (WB)**
    - Writes computed result back to register file

Pipeline registers (**IF/ID, ID/EX, EX/MEM, MEM/WB**) are used to store intermediate results between stages.

◆ Instruction Format (16-bit)

Bits	Field
15–12	Opcode
11–8	Source Register (rs)
7–4	Source Register (rt)
3–0	Destination Register (rd)

◆ Operation / Methodology

Opcode	Instruction	Operation
0000	ADD	rd = rs + rt
0001	SUB	rd = rs − rt
0010	LOAD	rd = data_mem[rt]

Execution Flow:

1. Instruction fetched using PC
2. Instruction decoded and passed forward
3. ALU executes operation
4. Result written back to register file

Each stage completes in **one clock cycle**, enabling pipelined execution.

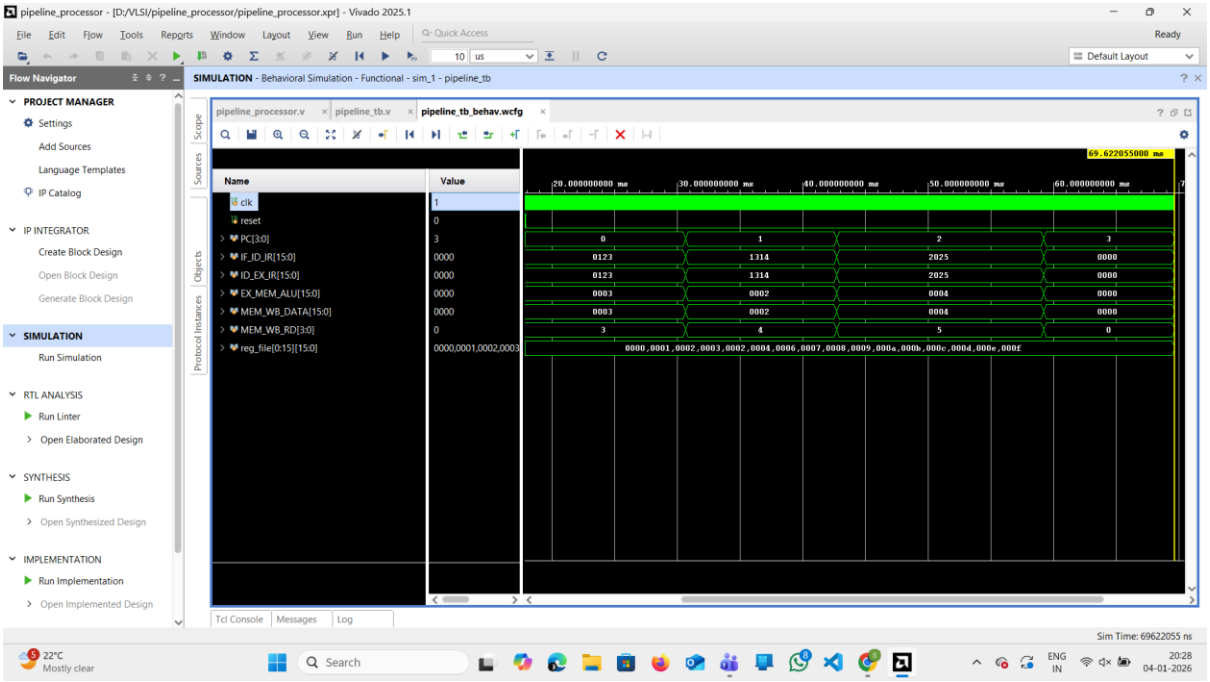
🔹 Simulation & Testbench

- A testbench was created to generate:
  - Clock signal
  - Reset signal
  - Instruction sequence
- Instruction memory and registers were initialized
- Simulation was run using **Vivado Behavioral Simulation**
- Internal signals and pipeline registers were observed in the waveform window

🔹 Simulation Results

- Instructions successfully moved through all pipeline stages
- Register values updated correctly after write-back
- Pipeline registers showed proper instruction flow
- Final register values matched expected outputs

📌 Observed Registers Example:



Note: R0=0, R1=1, R2=2, R3=3, R4=2, R5=4

Applications

- Educational processor design
- Understanding pipelining concepts
- RTL design practice
- FPGA and VLSI learning projects

Conclusion

The project successfully demonstrates a **4-stage pipelined processor** using Verilog HDL.
Simulation results confirm correct instruction execution and pipeline operation.
This design provides a strong foundation for understanding pipelined processor architecture.