Overview

This document provides an overview of a Verilog-based CPU module designed for a 19-bit instruction set architecture, along with a corresponding testbench to validate its functionality. The CPU supports arithmetic, logical, and control flow instructions, and is implemented using a simple pipeline architecture.

CPU Module

The CPU module processes 19-bit instructions and interacts with memory through mem_read and mem_write signals. It includes a register file with 16 registers, a basic ALU, and pipeline registers to handle different stages of instruction execution.

Inputs

- clk: Clock signal
- reset: Reset signal
- instruction: 19-bit instruction input
- data_in: 16-bit data input from memory

Outputs

- data_out: 16-bit data output to memory
- address: 16-bit memory address output
- mem_read: Memory read enable signal
- mem_write: Memory write enable signal

Instruction Format

The 19-bit instruction is divided into several fields:

- **Opcode (4 bits)**: Specifies the operation to be performed.
- **Register Source (rs, 4 bits)**: First source register.
- **Register Target (rt, 4 bits)**: Second source register.
- **Register Destination (rd, 4 bits)**: Destination register.
- **Function Code (func, 3 bits)**: Additional operation specification (used for certain instructions).
- **Address (12 bits)**: Used for control flow instructions (e.g., jump, branch).

Supported Instructions

Arithmetic Instructions

1. **ADD r1, r2, r3**: r1 = r2 + r3
2. **SUB r1, r2, r3**: r1 = r2 - r3
3. **MUL r1, r2, r3**: r1 = r2 * r3

4. **DIV r1, r2, r3**: r1 = r2 / r3
5. **INC r1**: r1 = r1 + 1
6. **DEC r1**: r1 = r1 - 1

Logical Instructions

1. **AND r1, r2, r3**: r1 = r2 & r3
2. **OR r1, r2, r3**: r1 = r2 | r3
3. **XOR r1, r2, r3**: r1 = r2 ^ r3
4. **NOT r1, r2**: r1 = ~r2

Control Flow Instructions

1. **JMP addr**: PC = addr
2. **BEQ r1, r2, addr**: if (r1 == r2) PC = addr
3. **BNE r1, r2, addr**: if (r1 != r2) PC = addr
4. **CALL addr**: store PC+1 in r15, PC = addr

Pipeline Stages

1. **Fetch Stage (IF)**: Instruction fetch from memory.
2. **Decode Stage (ID)**: Decode instruction and read registers.
3. **Execute Stage (EX)**: Perform ALU operations.
4. **Memory Stage (MEM)**: Read from/write to memory.
5. **Write-back Stage (WB)**: Write results back to registers.

ALU Operations

The ALU performs arithmetic and logical operations based on the decoded opcode and function code. Supported operations include addition, subtraction, multiplication, division, and bitwise operations (AND, OR, XOR, NOT).

Testbench

The testbench simulates the CPU module and verifies its functionality by applying a series of test instructions and checking the expected results.

Key Features

- Initializes the CPU module and applies reset.
- Provides a sequence of 19-bit instructions to test different operations.
- Checks the contents of registers and outputs after each instruction to verify correct execution.
- Displays error messages if any test fails.