Task 3: Pipeline Processor Design

This report presents the design of a 4-stage pipelined processor with basic instructions ADD, SUB, and LOAD. The design is based on the standard instruction pipeline stages:

1. IF - Instruction Fetch

2. ID - Instruction Decode

3. EX - Execute (ALU operations)

4. WB - Write Back

The pipeline allows overlapping execution of multiple instructions, improving throughput.

Pipeline Stages

1. Instruction Fetch (IF): Fetches the instruction from memory.

2. Instruction Decode (ID): Decodes the fetched instruction and reads registers.

3. Execute (EX): Performs the arithmetic operation (ADD/SUB) or address calculation for LOAD.

4. Write Back (WB): Writes the result back to the register file.

Instruction Set

1. ADD R1, R2, R3 : R1 <- R2 + R3

2. SUB R1, R2, R3 : R1 <- R2 - R3

3. LOAD R1, [R2] : R1 <- MEM[R2]

Example Pipeline Execution

Consider the instruction sequence: ADD R1,R2,R3; SUB R4,R1,R5; LOAD R6,[R4]

Cycle-by-cycle execution (simplified):

Cycle 1: IF(ADD)

Cycle 2: ID(ADD), IF(SUB)

Cycle 3: EX(ADD), ID(SUB), IF(LOAD)

Cycle 4: WB(ADD), EX(SUB), ID(LOAD)

Cycle 5: WB(SUB), EX(LOAD)

Cycle 6: WB(LOAD)

Verilog Code Snippet

```
module pipeline_processor(input clk, input reset);
  reg [31:0] IF_ID, ID_EX, EX_WB;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      IF_ID <= 0; ID_EX <= 0; EX_WB <= 0;
    end else begin
      // IF Stage
      IF_ID <= instruction;
      // ID Stage
      ID_EX <= decode(IF_ID);
      // EX Stage
      EX_WB <= execute(ID_EX);
      // WB Stage
      write_back(EX_WB);
    end
  end
endmodule
```