Simple Verilog Datapath: Design, Code, and Testbench

1. Problem Statement

Objective:

To design a simple datapath that performs arithmetic operations (addition and subtraction)

between two 8-bit inputs using basic combinational (MUX, ALU) and sequential (register) elements.

Specifications:

Inputs:

- inA, inB: 8-bit data inputs

- sel_mux: MUX select control

- op_alu: ALU operation control (0 for ADD, 1 for SUB)

- loadA, loadB, loadOut: Register load enables

- clk: Clock signal

Output:

- result: 8-bit ALU result stored in output register

Functional Description:

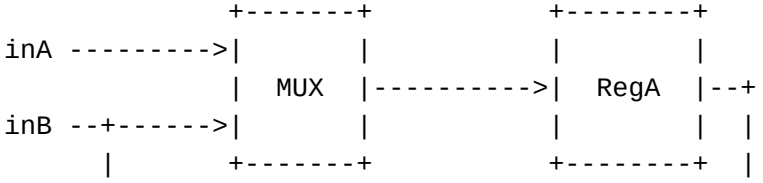
- A MUX selects between inA and inB and loads the result into Register A.

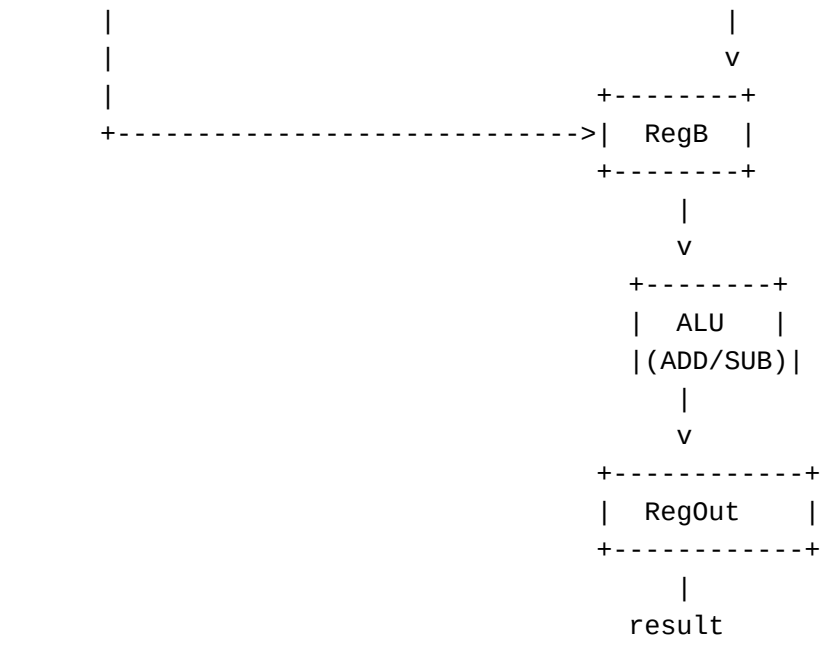
- inB is directly loaded into Register B.

- ALU performs add/sub between Register A and Register B.

- The result is stored in an output register.

2. Block Diagram Description





3. Verilog Code - mux2to1.v

```
module mux2to1(
    input [7:0] in0, in1,
    input sel,
    output [7:0] out
);
    assign out = sel ? in1 : in0;
endmodule
```

4. Verilog Code - alu.v

```
module alu(
    input [7:0] a, b,
    input op,
    output [7:0] result
);
    assign result = op ? (a - b) : (a + b);
endmodule
```

5. Verilog Code - register.v

```
module register(
    input clk,
    input [7:0] d,
    input en,
    output reg [7:0] q
);
    always @(posedge clk) begin
        if (en)
            q <= d;
    end
endmodule
```

6. Verilog Code - datapath.v

```
module datapath(
    input clk,
```

```
    input [7:0] inA, inB,
    input sel_mux,
    input op_alu,
    input loadA, loadB, loadOut,
    output [7:0] result
);
    wire [7:0] mux_out, regA_out, regB_out, alu_out;

    mux2to1 u_mux (.in0(inA), .in1(inB), .sel(sel_mux), .out(mux_out));
    register u_regA (.clk(clk), .d(mux_out), .en(loadA), .q(regA_out));
    register u_regB (.clk(clk), .d(inB), .en(loadB), .q(regB_out));
    alu u_alu (.a(regA_out), .b(regB_out), .op(op_alu), .result(alu_out));
    register u_regOut (.clk(clk), .d(alu_out), .en(loadOut), .q(result));
endmodule
```

7. Verilog Code - Testbench

```
module tb_datapath;
    reg clk = 0;
    reg [7:0] inA, inB;
    reg sel_mux, op_alu, loadA, loadB, loadOut;
    wire [7:0] result;

    datapath uut (
        .clk(clk), .inA(inA), .inB(inB), .sel_mux(sel_mux), .op_alu(op_alu),
        .loadA(loadA), .loadB(loadB), .loadOut(loadOut), .result(result)
    );

    always #5 clk = ~clk;

    initial begin
        inA = 8'd15; inB = 8'd5;
        sel_mux = 0; loadA = 1; loadB = 1; loadOut = 0; op_alu = 0;
        #10; loadA = 0; loadB = 0;

        op_alu = 0; loadOut = 1;
        #10; loadOut = 0;

        #10;
        $stop;
    end
endmodule
```