ALU Normal code

module pipelined\_alu #(parameter width = 32)(

input wire clk,

input wire rst,

input wire [width-1:0] a,

input wire [width-1:0] b,

input wire [2:0] ALUop,

output reg [width-1:0] result

);

// pipeline registers

reg [width-1:0] a\_reg1, b\_reg1, alu\_out\_reg2;

reg [2:0] ALUOP\_reg1, ALUOP\_reg2;

// stage 1 --> stage 2 --> Fetch/Decode

always @ (posedge clk or posedge rst)

begin

if(rst) begin

a\_reg1 <= 0;

b\_reg1 <= 0;

ALUOP\_reg1 <= 0;

end

else begin

a\_reg1 <= a;

b\_reg1 <= b;

ALUOP\_reg1 <= ALUop;

end

end

// stage 2 --> stage 3 --> Execution

always @ (posedge clk or posedge rst)

begin

if(rst) begin

alu\_out\_reg2 <= 0;

ALUOP\_reg2 <= 0;

end

else begin

ALUOP\_reg2 <= ALUOP\_reg1;

case(ALUOP\_reg1)

// add

3'b000 : alu\_out\_reg2 <= a\_reg1 + b\_reg1;

// sub

3'b001 : alu\_out\_reg2 <= a\_reg1 - b\_reg1;

// anding

3'b010 : alu\_out\_reg2 <= a\_reg1 & b\_reg1;

// oring

3'b011 : alu\_out\_reg2 <= a\_reg1 | b\_reg1;

// xoring

3'b100 : alu\_out\_reg2 <= a\_reg1 ^ b\_reg1;

// less than

3'b101 : alu\_out\_reg2 <= (a\_reg1 < b\_reg1) ? 1 : 0;

default : alu\_out\_reg2 <= 0;

endcase

end

end

// write back

always @ (posedge clk or posedge rst)

begin

if(rst) begin

result <= 0;

end

else begin

result <= alu\_out\_reg2;

ALUOP\_reg2 <= ALUOP\_reg1;

end

end

endmodule

ALU Testbench

module ALU\_tb;

// Inputs

reg clk;

reg rst;

reg [31:0] a;

reg [31:0] b;

reg [2:0] ALUop;

// Outputs

wire [31:0] result;

// Instantiate the Unit Under Test (UUT)

pipelined\_alu uut (

.clk(clk),

.rst(rst),

.a(a),

.b(b),

.ALUop(ALUop),

.result(result)

);

always #5 clk = ~clk;

task apply\_op(input [31:0] A, input [31:0] B, input [2:0]op);

begin

@(posedge clk)

a=A;

b=B;

ALUop=op;

repeat(2) @(posedge clk);

$display("Time=%0t , A=%0b , B=%0b , op = %b , result=%0b", $time,A,B,op,result);

end

endtask

initial begin

//$monitor("Time=%0t, a=%0d, b=%0d, ALUop = %b, result=%0d",$time, a, b, ALUop, result);

clk = 0;

a=0;

b=0;

ALUop=0;

rst = 1;

#10;

rst = 0;

@(posedge clk)

apply\_op(10,5,3'b000); // 10+5

#10;

//@(posedge clk)

apply\_op(20,8,3'b001); // 20-8

#10;

//@(posedge clk)

apply\_op(32'hFF00FF00, 32'h0F0F0F0F, 3'b010); // anding

#10;

//@(posedge clk)

apply\_op(32'hFF00FF00, 32'h0F0F0F0F, 3'b011); // oring

#10;

//@(posedge clk)

apply\_op(0,1,3'b100); // xoring

#10;

//@(posedge clk)

apply\_op(3,7,3'b101); // less than

#10;

#20;

$stop;

end

endmodule