module ALU(

input [8:0] A, B, // Input registers (A is Num 1, B is Num 2)

input [3:0] select, // Select bits

output [8:0] out // Output

);

reg [8:0] result; // Result register

assign out=result; // Assign arithmetic or logical result to output register

always @(\*)

begin

case(select)

4'b0000: // Opcode for Add

result=(A+B);

4'b0001: // Opcode for Sub

result=(A-B);

4'b0010: // Opcode for left shift

result=(A<<1); // (Note that only A (Num 1) is operated on, thus B (Num 2) has no effect

4'b0011: // Opcode for shift right

result=(A>>1); // (Note that only A (Num 1) is operated on, thus B (Num 2) has no effect

4'b0100: // Opcode for AND

result=(A&B);

4'b0101: // Opcode for OR

result=(A|B);

4'b0110: // Opcode for XOR

result=(A^B);

4'b0111: // Opcode for NOT

result=(~A); // (Note that only A (Num 1) is operated on, thus B (Num 2) has no effect

endcase

end

endmodule

module testbench1;

reg[8:0] A, B; // Input registers (A is Num 1, B is Num 2)

reg[3:0] select; // Select bits

wire[8:0] out; // Output

integer i; // For loop counter

ALU test1(

A, B, // Input registers (A is Num 1, B is Num 2)

select, // Selection bits

out // Output

);

initial begin

A = 8'h01; // First input (Num 1)

B = 8'hFF; // Second input (Num 2)

select = 4'h0; // Initialize select bits to all zeros

// Display labels

$display("Num 1 Num 2 Operation Current Output Next State ");

// Loop for displaying Num 1, Num 2, Operation, and Current State

for (i=0;i<=8;i=i+1)

begin

// Ensure output is within valid range

if ((out>=9'b000000000) && (out<=9'b111111111))

// Display Num 1, Num 2, and Opcode

$write("%b", A, " (", "%d", A, ") ", "%b", B, " (", "%d", B, ") ", "%b", (select-1'b1), " (");

// Display Operation depending on select bits and Current State

case(select)

4'b0001:

$write("Add) Running ");

4'b0010:

$write("Sub) Running ");

4'b0011:

$write("Left) Running ");

4'b0100:

$write("Right) Running ");

4'b0101:

$write("AND) Running ");

4'b0110:

$write("OR) Running ");

4'b0111:

$write("XOR) Running ");

4'b1000:

$write("NOT) Running ");

endcase

// Output validation and error handling for Add, Left Shift

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i==1) || (i==3)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("XXXXXXXXX (Nan) ERROR");

end

// Output validation and error handling for Sub

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && (i==2) )

begin

if (out>=9'b100000000)

$display("%b", out, " (-", "%d", ~(out-1'b1), ") Running");

else

$display("%b", out, " (", "%d", out, ") Running");

end

// Output validation and error handling for Right Shift, AND, OR, XOR, NOT

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i>=4) && (i<=8)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("%b", (out-9'b100000000), " (", "%d", (out-9'b100000000), ") Running");

end

#10;

select=select+4'h1; // Add 1 to select bits

end

$display("");

end

endmodule

module testbench2;

reg[8:0] A, B; // Input registers (A is Num 1, B is Num 2)

reg[3:0] select; // Select bits

wire[8:0] out; // Output

integer i; // For loop counter

ALU test2(

A, B, // Input registers (A is Num 1, B is Num 2)

select, // Selection bits

out // Output

);

initial begin

A = 8'hEE; // First input (Num 1)

B = 8'hA0; // Second input (Num 2)

select = 4'h0; // Initialize select bits to all zeros

// Loop for displaying Num 1, Num 2, Operation, and Current State

for (i=0;i<=8;i=i+1)

begin

// Ensure output is within valid range

if ((out>=9'b000000000) && (out<=9'b111111111))

// Display Num 1, Num 2, and Opcode

$write("%b", A, " (", "%d", A, ") ", "%b", B, " (", "%d", B, ") ", "%b", (select-1'b1), " (");

// Display Operation depending on select bits and Current State

case(select)

4'b0001:

$write("Add) Running ");

4'b0010:

$write("Sub) Running ");

4'b0011:

$write("Left) Running ");

4'b0100:

$write("Right) Running ");

4'b0101:

$write("AND) Running ");

4'b0110:

$write("OR) Running ");

4'b0111:

$write("XOR) Running ");

4'b1000:

$write("NOT) Running ");

endcase

// Output validation and error handling for Add, Left Shift

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i==1) || (i==3)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("XXXXXXXXX (Nan) ERROR");

end

// Output validation and error handling for Sub

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && (i==2) )

begin

if (out>=9'b100000000)

$display("%b", out, " (-", "%d", ~(out-1'b1), ") Running");

else

$display("%b", out, " (", "%d", out, ") Running");

end

// Output validation and error handling for Right Shift, AND, OR, XOR, NOT

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i>=4) && (i<=8)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("%b", (out-9'b100000000), " (", "%d", (out-9'b100000000), ") Running");

end

#100;

select=select+4'h1; // Add 1 to select bits

end

$display("");

end

endmodule

module testbench3;

reg[8:0] A, B; // Input registers (A is Num 1, B is Num 2)

reg[3:0] select; // Select bits

wire[8:0] out; // Output

integer i; // For loop counter

ALU test3(

A, B, // Input registers (A is Num 1, B is Num 2)

select, // Selection bits

out // Output

);

initial begin

A = 8'h70; // First input (Num 1)

B = 8'h0B; // Second input (Num 2)

select = 4'h0; // Initialize select bits to all zeros

// Loop for displaying Num 1, Num 2, Operation, and Current State

for (i=0;i<=8;i=i+1)

begin

// Ensure output is within valid range

if ((out>=9'b000000000) && (out<=9'b111111111))

// Display Num 1, Num 2, and Opcode

$write("%b", A, " (", "%d", A, ") ", "%b", B, " (", "%d", B, ") ", "%b", (select-1'b1), " (");

// Display Operation depending on select bits and Current State

case(select)

4'b0001:

$write("Add) Running ");

4'b0010:

$write("Sub) Running ");

4'b0011:

$write("Left) Running ");

4'b0100:

$write("Right) Running ");

4'b0101:

$write("AND) Running ");

4'b0110:

$write("OR) Running ");

4'b0111:

$write("XOR) Running ");

4'b1000:

$write("NOT) Running ");

endcase

// Output validation and error handling for Add, Left Shift

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i==1) || (i==3)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("XXXXXXXXX (Nan) ERROR");

end

// Output validation and error handling for Sub

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && (i==2) )

begin

if (out>=9'b100000000)

$display("%b", out, " (-", "%d", ~(out-1'b1), ") Running");

else

$display("%b", out, " (", "%d", out, ") Running");

end

// Output validation and error handling for Right Shift, AND, OR, XOR, NOT

if ( ((out>=9'b000000000) && (out<=9'b111111111)) && ((i>=4) && (i<=8)) )

begin

if (out<=9'b011111111)

$display("%b", out, " (", "%d", out, ") Running");

else

$display("%b", (out-9'b100000000), " (", "%d", (out-9'b100000000), ") Running");

end

#1000;

select=select+4'h1; // Add 1 to select bits

end

end

endmodule