Table 1

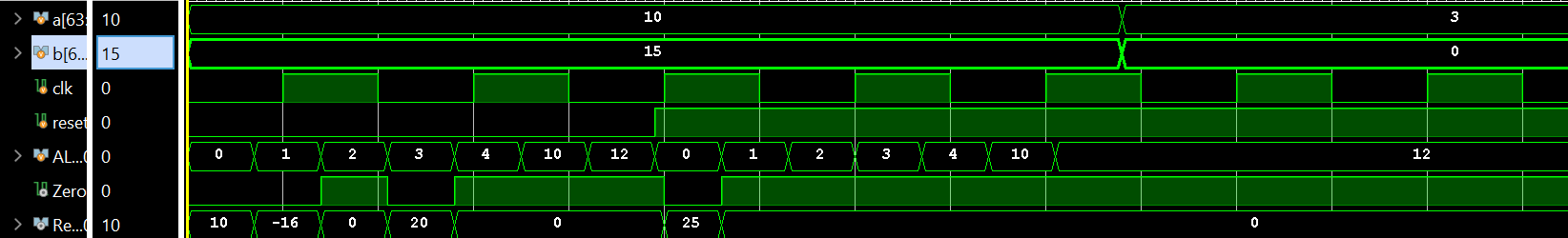
Rohan Raj

07656

PAPER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Reset** | **CLK Value** | **ALU function** | **a** | **a** | **Zero1** | **Result** |
| 1 | 0 | 1 | 0 | 10 | 15 | 0 | 10 |
| 2 | 0 | 1 | 1 | 10 | 15 | 0 | -16 |
| 3 | 0 | 1 | 2 | 10 | 15 | 1 | 0 |
| 4 | 0 | 1 | 3 | 10 | 15 | 0 | 20 |
| 5 | 0 | 1 | 4 | 10 | 15 | 1 | 0 |
| 6 | 0 | 1 | 10 | 10 | 15 | 1 | 0 |
| 7 | 0 | 1 | 12 | 10 | 15 | 1 | 0 |
| 8 | 1 | 1 | 0 | 10 | 15 | 1 | 0 |
| 9 | 1 | 1 | 1 | 10 | 15 | 1 | 0 |
| 10 | 1 | 1 | 2 | 10 | 15 | 1 | 0 |
| 11 | 1 | 1 | 3 | 10 | 15 | 1 | 0 |
| 12 | 1 | 1 | 4 | 10 | 15 | 1 | 0 |

# Timing Diagram 1



# Timing Diagram 2

# Codes:

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 03/21/2023 03:43:59 PM

// Design Name:

// Module Name: ALU

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 03/21/2023 03:43:59 PM

// Design Name:

// Module Name: ALU

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 03/21/2023 03:43:59 PM

// Design Name:

// Module Name: ALU

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module ALU(

input [63:0] a,

input [63:0] b,

input clk, reset,

input [3:0] ALUOp,

output Zero,

output reg [63:0] result

);

always @ (\*)

begin

if (reset)

begin

assign result = 0;

end

else

begin

// case (ALUOp)

if (ALUOp == 4'b0000)

assign result = a & b; // Bitwise AND operation

else if (ALUOp == 4'b0001)

assign result = ~(a | b); // Bitwise NOR operation

else if (ALUOp == 4'b0010)

assign result = a > b;

else if (ALUOp == 4'b0011)

assign result = a \* 2;

else if (ALUOp == 4'b0100)

begin

assign result = a==b ;

end

else if (ALUOp == 4'b1010)

begin @(posedge clk)

assign result = a + b; // Addition operation

end

else if (ALUOp == 4'b1100)

begin @(posedge clk)

assign result = a - b; // Subtraction operation

end

end

end

// Output ZERO if result is equal to zero, otherwise output 0

assign Zero = (result == 64'b0) ? 1'b1 : 1'b0;

endmodule

Test Bench:

`timescale 1ns / 1ps

module ALU\_tb();

reg [63:0] a; // register for input operand a

reg [63:0] b; // register for input operand b

reg clk, reset;

reg [3:0] ALUOp; // register for control signal

wire Zero; // wire for zero flag

wire [63:0] Result; // wire for output result

ALU ALU1(.a(a), .b(b), .clk(clk), .reset(reset), .ALUOp(ALUOp), .Zero(Zero), .result(Result)); // instantiating ALU module

initial

begin

a = 64'd10; // assigning decimal value 30 to operand a

b = 64'd15; // assigning decimal value 20 to operand b

clk = 1'b0;

reset = 1'b0;

ALUOp = 4'b0000; // assigning AND operation

#7

ALUOp = 4'b0001; // assigning NOR operation

#7

ALUOp = 4'b0010;

#7

ALUOp = 4'b0011;

#7

ALUOp = 4'b0100;

#7

ALUOp = 4'b1010; // assigning Add operation

#7

ALUOp = 4'b1100; // assigning Subtract operation

#7

reset = 1;

ALUOp = 4'b0000; // assigning AND operation

#7

ALUOp = 4'b0001; // assigning NOR operation

#7

ALUOp = 4'b0010;

#7

ALUOp = 4'b0011;

#7

ALUOp = 4'b0100;

#7

ALUOp = 4'b1010; // assigning Add operation

#7

ALUOp = 4'b1100; // assigning Subtract operation

#7

a = 64'd3; // assigning binary value 1 to operand a

b = 64'd0; // assigning binary value 0 to operand b

end

always

#10 clk = ~clk;

endmodule