VLSI System Design

ELE301P

LAB - 7 - Report

Praveen B R

COE19B007

Submission Date: 27/10/2021

INDEX

Q1)

Objective:

To design and implement an 8bit Arithmetic Logic Unit (ALU) in Verilog with the given operations.

Theory:

An ALU (Arithmetic Logic Unit) is a digital electronic circuit which conducts operations on binary numbers in arithmetic and bitwise logical ways. The latter operates on binary numbers in terms of individual bits whilst numbers are covered through the arithmetic processes.

For the given assignment I decided to design an ALU for 8 bit input numbers to perform given tasks based on select line.

Code:

```
module alu8(a, b, sel, out, cf, zf, nf, pf, of);
    input [7:0] a, b;
    input [3:0] sel;
    output cf, zf, nf, pf, of;
    reg zf, pf, of;
    output [7:0] out;
    integer count;
    reg [8:0] a_reg;
    reg [8:0] result;
    wire temp;
    assign out = result[7:0];
    assign cf = result[8];
    assign nf = result[7];

    always @(*)
    begin
        zf = 1'b0;
        pf = 1'b0;
        of = 1'b0;
        case (sel)
            4'd0:
                result = a + b; // Addition
            4'd1:
                result = a - b; // Subtraction
            4'd2:
                result = a * b; // Multiplication
            4'd3:
```

```
            result = a / b; // Division
        4'd4:
            result = a & b; // Bitwise AND
        4'd5:
            result = a | b; // Bitwise OR
        4'd6:
            result = a ^ b; // Bitwise XOR
        4'd7:
            result = ~(a | b); // Bitwise NAND
        4'd8:
            result = ~(a & b); // Bitwise NOR
        4'd9:
            result = ~(a ^ b); // Bitwise XNOR
        4'd10:
            result = a << b; // Left Shift
        4'd11:
            result = a >> b; // Right shift
        4'd12:
            begin   // Rotate Left with Carry
                a_reg = a;
                for(integer i = 0; i < b; i++)
                begin
                    a_reg = {a_reg[7:0], a_reg[8]};
                end
                result = a_reg;
            end
        4'd13:
            begin   // Rotate Right with Carry
                a_reg = {a[7:0], 1'b0};
                for(integer i = 0; i < b; i++)
                begin
                     a_reg = {a_reg[0], a_reg[8:1]};
                end
                a_reg = {a_reg[0], a_reg[9:1]};
                result = a_reg;
            end
        4'd14:
            result = (a == b) ? 8'd1 : 8'd0;
    endcase
```

```
        if(out == 8'd0)
            zf = 1'b1;

        count = 1'b0;
        for(integer j = 0; j < 8; j++) begin
            if(result[j] == 1'b1)
                count++;
        end
        if(count % 2 == 0)
            pf = 1'b1;

        if(a[7] == b[7]) begin
            if(result[7] != a[7])
                of = 1'b1;
        end
    end

endmodule

module alu8_tb;
    reg[7:0] a, b;
    reg[3:0] sel;

    wire[7:0] out;
    wire cf, zf, nf, pf, of;

    alu8 alu1(a, b, sel, out, cf, zf, nf, pf, of);
    initial begin
        a = 8'b10101010;
        b = 8'b11110000;
    end
    initial begin
        #0      sel=4'b0000;
        #14     $finish;
    end
    always begin
    #1 sel += 1;
    end
    initial begin
        $dumpfile("alu8.vcd");
```

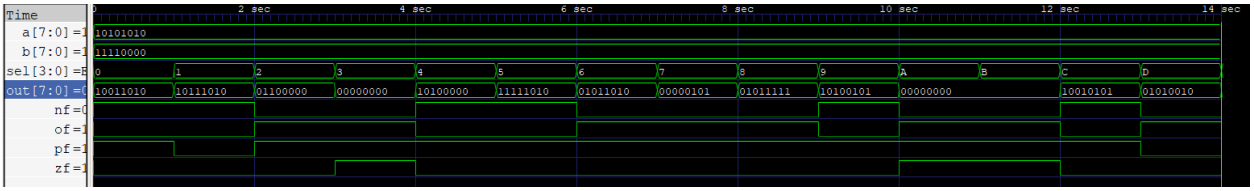
```
        $dumpvars(0,alu8_tb);
        $monitor("  a=%b   b=%b   sel=%d   out=%b   carry=%b zf=%b nf=%b
pf=%b of=%b", a, b, sel, out, cf, zf, nf, pf, of);
    end

endmodule
```

Output/Waveform:

Terminal

```
PS E:\Sem 5\VLSI\Lab\lab7> iverilog 8alu.v
PS E:\Sem 5\VLSI\Lab\lab7> vvp a.out
VCD info: dumpfile alu8.vcd opened for output.
  a=11110000   b=10101010   sel= 0   out=10011010   carry=1 zf=0 nf=1 pf=1 of=0
  a=11110000   b=10101010   sel= 1   out=01000110   carry=0 zf=0 nf=0 pf=0 of=1
  a=11110000   b=10101010   sel= 2   out=01100000   carry=1 zf=0 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel= 3   out=00000001   carry=0 zf=0 nf=0 pf=0 of=1
  a=11110000   b=10101010   sel= 4   out=10100000   carry=0 zf=0 nf=1 pf=1 of=0
  a=11110000   b=10101010   sel= 5   out=11111010   carry=0 zf=0 nf=1 pf=1 of=0
  a=11110000   b=10101010   sel= 6   out=01011010   carry=0 zf=0 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel= 7   out=00000101   carry=1 zf=0 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel= 8   out=01011111   carry=1 zf=0 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel= 9   out=10100101   carry=1 zf=0 nf=1 pf=1 of=0
  a=11110000   b=10101010   sel=10   out=00000000   carry=0 zf=1 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel=11   out=00000000   carry=0 zf=1 nf=0 pf=1 of=1
  a=11110000   b=10101010   sel=12   out=11000011   carry=0 zf=0 nf=1 pf=1 of=0
  a=11110000   b=10101010   sel=13   out=00111100   carry=0 zf=0 nf=0 pf=1 of=1
8alu.v:108: $finish called at 14 (1s)
  a=11110000   b=10101010   sel=14   out=00000000   carry=0 zf=1 nf=0 pf=1 of=1
```

Waveform



Conclusion:

Thus, an 8bit Arithmetic Logic Unit (ALU) has been designed and implemented in Verilog with the given operations.

Application:

ALU is stands for **arithmetic and logical unit** which performs all arithmetic operations like **addition**, **subtraction**, **division** and **multiplication** and **logical operations** of the computer system. It works the coordination of **control unit** and **memory uni**t. It is the **building block** of CPU. Modern CPU contains strong and most complex ALUs and modern CPUs holds a control unit

Miscellaneous Questions

Question 1 Explain the following data types in verilog

a) WAND

Wand is a net type variable which can take multiple values and perform AND operation on them and give the Output.

b) WOR

Wor is a net type variable which can take multiple values and perform OR operation on them and give the Output.

Code for WAND,WOR

```
module Type_wand(A,y);
input A;
output y;
wand k;
supply0 B;
assign k = A & B;
assign k = A | B;
assign y = k;
endmodule

module Type_wor(A,y);
input A;
output y;
wor k;
supply1 B;
assign k = A & B;
assign k = A | B;
assign y = k;
endmodule

module oq_tb;
reg A,B;
wire x,y;
Type_wor t1(A,x);
Type_wand t2(B,y);
initial
begin
A = 1;
```

```
B = 1;
end
initial
begin
$monitor("WOR    A=%b x=%b\nWAND   B=%b y=%b",A,x,B,y);
end
endmodule
```

c) TRI

TRI statement used to for assign different values to a same variable based on the state of assigning values

```
module Type_tri(A,B,y);
input A,B;
output y;
tri z;
assign z = A;
assign z = B;
assign y = z;
endmodule
module test;
reg A,B;
wire y;
Type_tri t1(A,B,y);
initial
begin
#10 A = 1;B= 1'bz;
#10 A = 1'bz ;B =0;
end
initial
begin
$monitor("A = %b B =%b y = %b",A,B,y);
end
endmodule
```

d) Supply0

Supply0 are the wires tied to ground or logical value 0.ada

e) Supply1

Supply1 are the wires tied to power or logical value 1.

```
module supply_type(Output);
```

```
    output Output;
    supply0 A;
    supply1 B;

    assign Output = A | B;
endmodule


module supply_tb();
    wire Output;

    supply_type S1 (Output);

    initial begin
        $monitor ("Output=%b",Output);
    end
endmodule
```

<u>Question 2</u> What is task statement?

- ☐ Task is a general statement and can calculate multiple result values and return them using **output** and **input** type arguments.
- ☐ Tasks can contain time-consuming simulation elements such as @, posedge, and others. Tasks are used in all programming languages, generally known as procedures or subroutines.
- ☐ Included in the main body of code, they can be called many times, reducing code repetition

```
module q3_tb;
reg [2:0] A,B,Sum;
task add;
input [2:0] A;
input [2:0] B;
output [2:0] Sum;
begin
Sum = A + B;
end
endtask
initial
begin
```

```
A = 3'b100;B = 3'b001;
add(A,B,Sum);
end
initial
begin
$monitor("A = %b = %b Sum=%b",A,B,Sum);
end
endmodule
```