COAL Lab # 02

ALU Designing in Verilog

Name: Muhammad Usman

Class: BSCS 3C1

Reg No: cs211208

Lab Exercise:

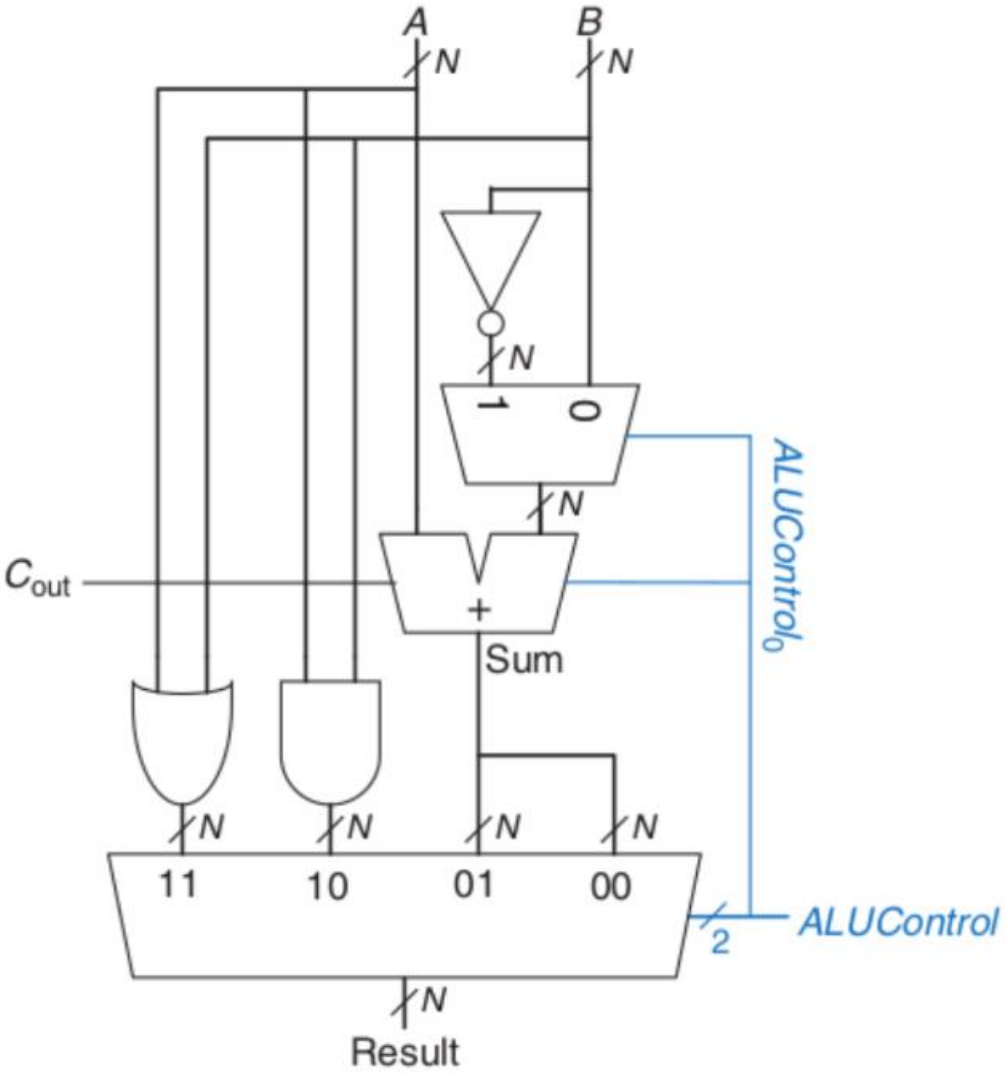
Literature Review:

An arithmetic-logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operands.ALU is a combinational circuit and the part of CPU that carries out arithmetic such as addition, subtraction, multiplication and logical operations such as AND, OR, NOT operations on the operands as per computer instructions.Multiplexer is being used in this circuit to select the specific output on the final result.

Task:

Write a Verilog module for the above ALU unit. Make sure all input and output signals have a unique name. Make a test bench to test all the operations by taking inputs of your own choices. Attach the test bench and the output waveforms.

Block Diagram:



Boolean Equation:

A_and_B = A & B;

A_or_B = A | B;

A_sum_B = A + B + control[0];

A_diff_B = A + ~(B) + control[0];

Verilog Code:

design.v

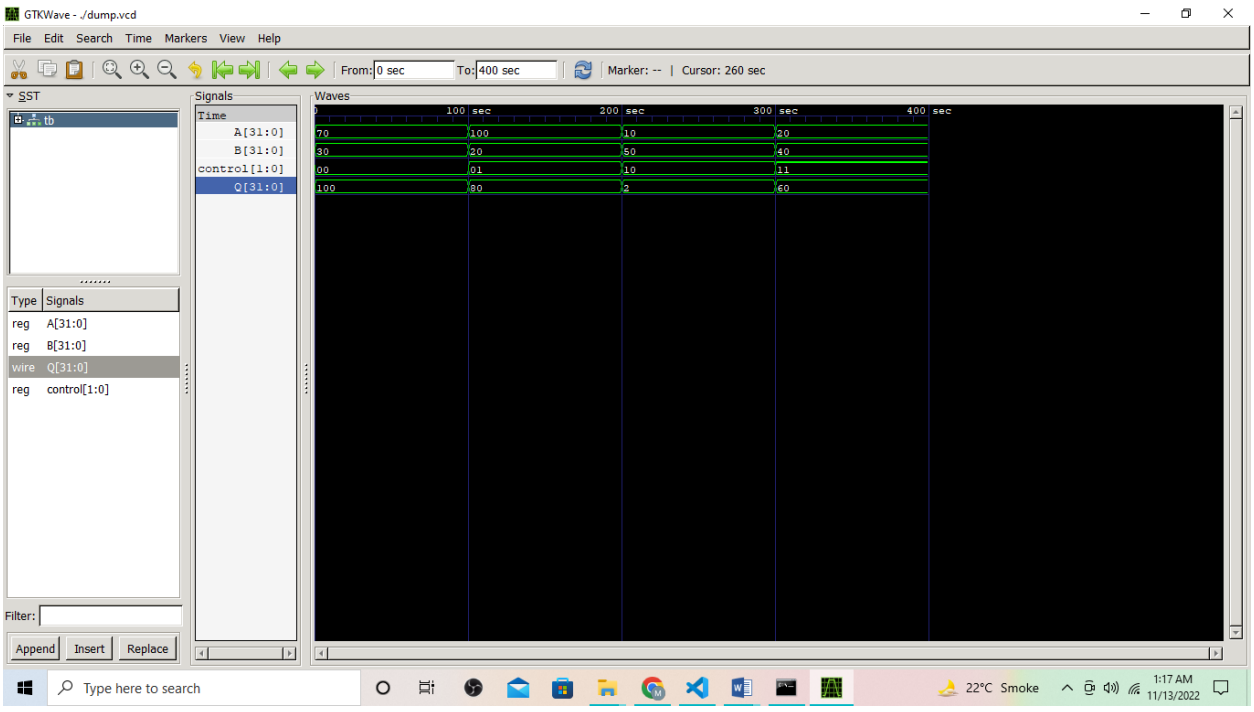
```
module ALU(A,B,Q,control);

    input [31:0]A,B;
    input[1:0]control;
    wire Cout;
    output [31:0]Q;
    wire [31:0]A_or_B,A_and_B,B_not,A_sum_B;
    wire [31:0]S1;

    assign A_or_B = A | B;
    assign A_and_B = A & B;
    assign B_not = ~B;
    assign S1 = (control[0] == 1'b1) ? B_not : B;
    assign {Cout,A_sum_B} = A+S1+control[0];
    assign Q = (control == 2'b00) ? A_sum_B : (control == 2'b01) ? A_sum_B :
(control == 2'b10) ? A_and_B : A_or_B;



endmodule
```

testbench.v

```
module tb();

    reg [31:0]A,B;
    reg [1:0]control;
    wire [31:0]Q;

    ALU dut (.A(A),.B(B),.Q(Q),.control(control));

    initial begin
      $dumpfile("dump.vcd");
      $dumpvars(0);
    end

    initial begin
        control <= 2'b00;
```

```
        A <= 32'd70;
        B <= 32'd30;
        #100
        control <= 2'b01;
        A <= 32'd100;
        B <= 32'd20;
        #100
        control <= 2'b10;
        A <= 32'd10;
        B <= 32'd50;
        #100
        control <= 2'b11;
        A <= 32'd20;
        B <= 32'd40;
        #100

        $finish;
    end


endmodule
```

Waveforms:



Post Lab Task:

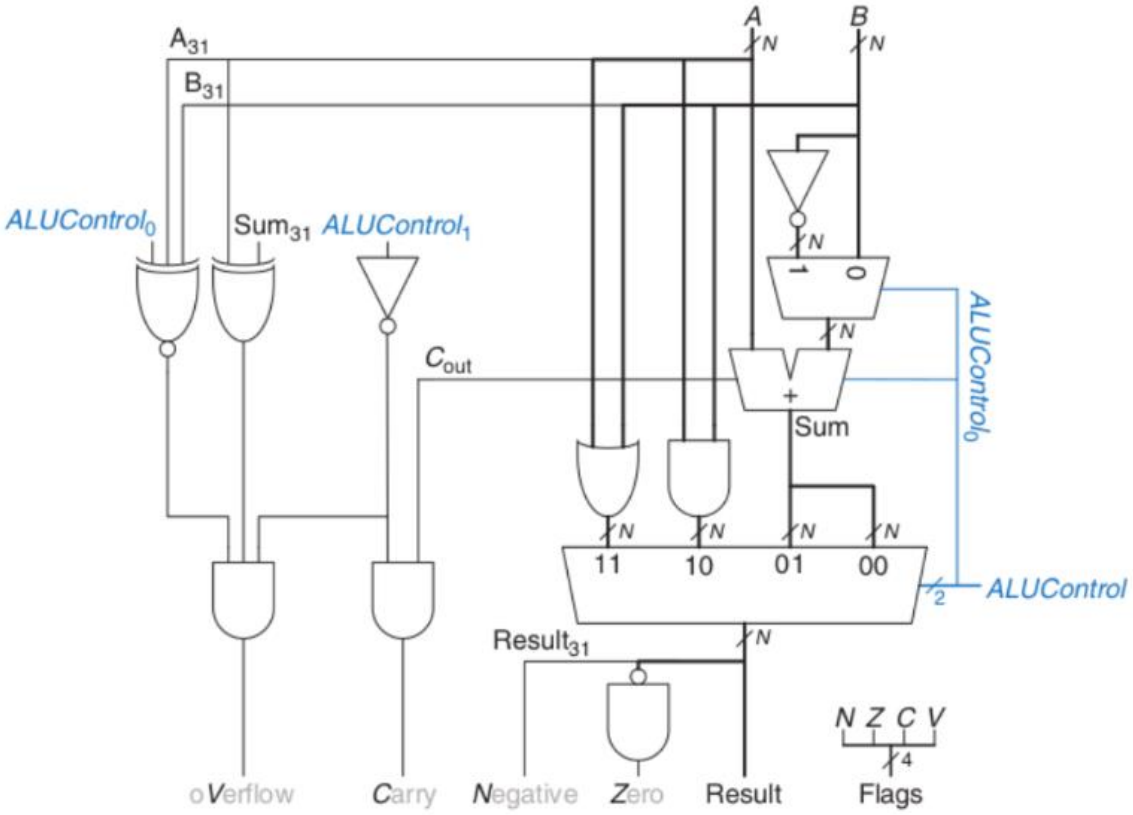
Literature Review:

An arithmetic-logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operands.ALU is a combinational circuit and the part of CPU that carries out arithmetic such as addition, subtraction, multiplication and logical operations such as AND, OR, NOT operations on the operands as per computer instructions.Multiplexer is being used in this circuit to select the specific output on the final result with Four flag outputs with one bits whose purpose is to show to true or false. Zero flag will produce logic level high, if output is zero. Negative flag will produce logic level high, if MSB of result is high. Carry flag will produce logic level high, if carry bit will have logic level high. Overflow condition is when specified output bits cannot store the produced output, so an overflow flag will generate logic level high.

Task:

Write a Verilog module for the given ALU figure. Make a test bench to test all the operations by taking inputs of your own choices. Attach the test bench and the output waveforms.

Block Diagram:



Boolean Equation:

A_and_B = A & B;

A_or_B = A | B;

A_sum_B = A + B + control[0];

A_diff_B = A + ~(B) + control[0];

result = Final Output;

Zero = &(~result);

Negative = result[31];

Verflow = ~(control[0] ^ A[31] ^ B[31]) & (A[31] ^ A_sum_B [31]) & (~control[1]);

Carry = ~control[1] & Cout;

Verilog Code:

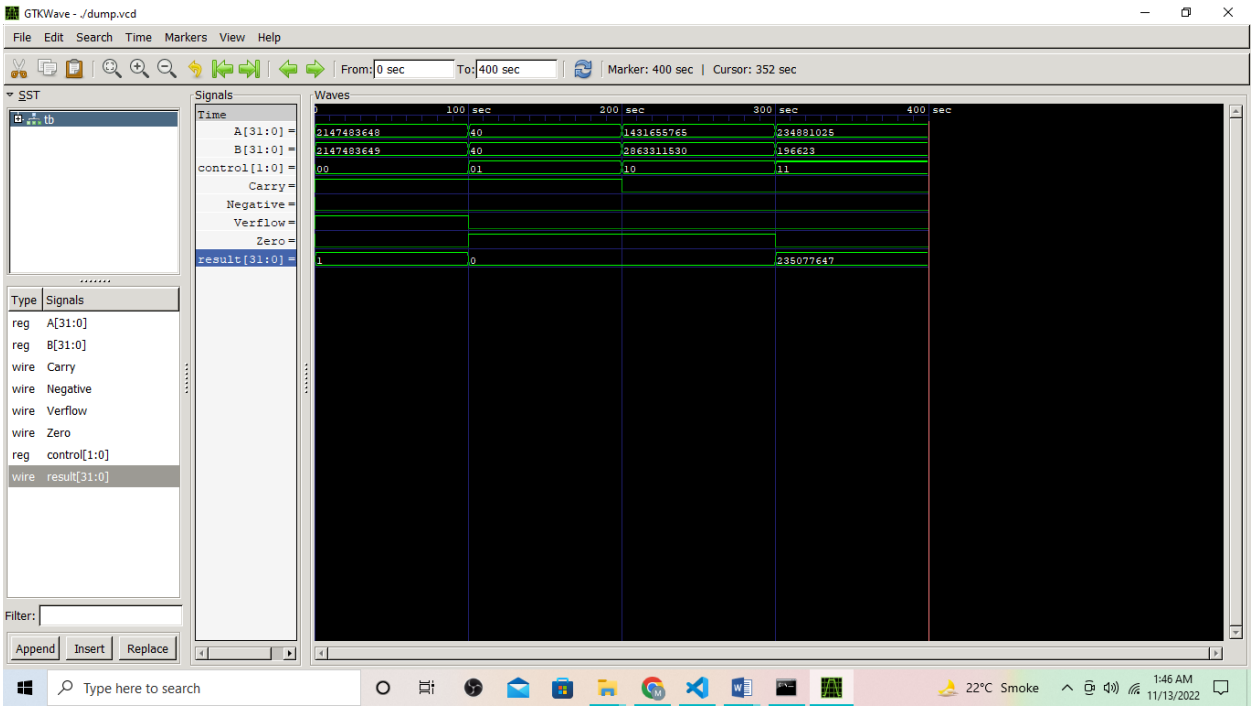
design.v

```
module ALU(A,B,result,control,Verflow,Carry,Negative,Zero);

    input [31:0]A,B;
    input[2:0]control;

    output [31:0]result;
    output Verflow,Carry,Negative,Zero;

    wire Cout;
    wire [31:0]A_or_B,A_and_B,B_not,A_sum_B;
    wire A_xor_sum,A_xnor_B_xnor_control,control_1_not;
    wire [31:0]S1;

    // Interim Outputs
    assign A_or_B = A | B;
    assign A_and_B = A & B;
    assign B_not = ~B;
    assign S1 = (control[0] == 1'b1) ? B_not : B;
    assign {Cout,A_sum_B} = A+S1+control[0];
    assign A_xor_sum = A[31]^A_sum_B[31];
    assign A_xnor_B_xnor_control = ~(A[31]^B[31]^control[0]);
    assign control_1_not = ~(control[1]);
    // Flags
    assign Verflow = A_xnor_B_xnor_control & A_xor_sum & control_1_not;
    assign Zero = &(~result);
    assign Negative = result[31];
    assign Carry = control_1_not & Cout;
    // Final Output
    assign result = (control[1:0] == 2'b00) ? A_sum_B : (control[1:0] == 2'b01) ?
A_sum_B : (control[1:0] == 2'b10) ? A_and_B : A_or_B ;


endmodule
```

testbench.v

```
module tb();

    reg [31:0]A,B;
    reg [1:0]control;
    wire [31:0]result;
    wire Verflow,Carry,Negative,Zero;

    ALU dut
(.A(A),.B(B),.result(result),.control(control),.Verflow(Verflow),.Carry(Carry),.N
egative(Negative),.Zero(Zero));

    initial begin
      $dumpfile("dump.vcd");
      $dumpvars(0);
    end

    initial begin
        control <= 2'b00;
        A <= 32'b10000000000000000000000000000000;
        B <= 32'b10000000000000000000000000000001;
        #100
        control <= 2'b01;
        A <= 32'd40;
        B <= 32'd40;
        #100
        control <= 2'b10;
        A <= 32'b01010101010101010101010101010101;
        B <= 32'b10101010101010101010101010101010;
        #100
        control <= 2'b11;
        A <= 32'b00001110000000000000000000000001;
        B <= 32'b00000000000000110000000000001111;
        #100
        $finish;
    end


endmodule
```

Waveforms:


GTKWave - ./dump.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 400 sec
Marker: 400 sec | Cursor: 352 sec
SST
tb
Type Signals
reg A[31:0]
reg B[31:0]
wire Carry
wire Negative
wire Verflow
wire Zero
reg control[1:0]
wire result[31:0]
Filter:
Append Insert Replace
Signals
Time
A[31:0] =
B[31:0] =
control[1:0] =
Carry =
Negative =
Verflow =
Zero =
result[31:0] =
Waves
100 sec
200 sec
300 sec
400 sec
2147483648
40
1431655765
234881025
2147483649
40
2863311530
196623
00
01
10
11
1
0
235077647
Type here to search
22°C Smoke
1:46 AM
11/13/2022