Vincent Li

Assignment 3

1)

Code:

```
// Filename: alu32.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    32-bit ALU with inputs A and B, output D,
    4-bit select input S, Cin, and Cout
*/

module alu32 #(parameter integer WIDTH = 32) (
    input [3 : 0] S,
    input [WIDTH - 1 : 0] A, B,
    input Cin,
    output reg [WIDTH - 1 : 0] D,
    output reg Cout
    );

    reg [WIDTH : 0] d;

    always @(S, A, B, Cin) begin
        if(S == 4'b0000) begin
            // do nothing
        end
        else if(S == 4'b0001) begin
            d = A + B + Cin;
        end
        else if(S == 4'b0010) begin
            d = A - B + Cin;
        end
        else if(S == 4'b0011) begin
            d = A * B;
        end
        else if(S == 4'b0100) begin
            d = A / B;
        end
        else if(S == 4'b0101) begin
            d = A << 1;
            d[0] = Cin;
        end
```

```
        else if(S == 4'b0110) begin
            d = A >> 1;
        end
        else if(S == 4'b0111) begin
            d = A << 1;
            d[0] = d[WIDTH];
            d[WIDTH] = 0;
        end
        else if(S == 4'b1000) begin
            d[WIDTH] = d[0];
            d = A >> 1;
        end
        else if(S == 4'b1001) begin
            d = A & B;
        end
        else if(S == 4'b1010) begin
            d = A | B;
        end
        else if(S == 4'b1011) begin
            d = A ^ B;
        end
        else if(S == 4'b1100) begin
            d = ~A + ~B;
        end
        else if(S == 4'b1101) begin
            d = ~A & ~B;
        end
        else if(S == 4'b1110) begin
            if(A < B) d = 0;
            else begin
                d[WIDTH : 1] = 0;
                d[0] = 1;
            end
        end
        else if(S == 4'b1111) begin
            if(~A == B) d = 0;
            else begin
                d[WIDTH : 1] = 0;
                d[0] = 1;
            end
        end
    end

    always @(d) begin
        D = d[WIDTH - 1 : 0];
```

```
        Cout = d[WIDTH];
    end

endmodule
```

```
// Filename: alu32_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    Test bench for alu32.v
*/

module alu32_tb;
    parameter integer WIDTH = 32;
    reg [WIDTH - 1 : 0] A, B;
    reg [3 : 0] S;
    reg Cin;
    wire [WIDTH - 1 : 0] D;
    wire Cout;

    integer allCorrect = 1;

    alu32 #(.WIDTH(WIDTH)) ALU (.A(A), .B(B), .S(S), .Cin(Cin), .D(D),
.Cout(Cout));

    initial
        begin
            S = 4'b0000;

            A = 32'b10101010101010101010101010101010;
            B = 32'b01010101010101010101010101010101;
            Cin = 0;
            #1;
        end
    initial
        begin

            repeat(16) begin

                // #1 $display("S = %b\nA = %b\nB = %b\nCin = %b\nD = %b\tCout =
%b\n",S, A, B, Cin, D, Cout);
                S = S + 1;
                #1;
```

```
                if(S == 4'b0001) begin
                    if(D != 32'b11111111111111111111111111111111 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0010) begin
                    if(D != 32'b01010101010101010101010101010101 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0011) begin
                    if(D != 32'b01110001110001110001110001110010 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0100) begin
                    if(D != 32'b00000000000000000000000000000010 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0101) begin
                    if(D != 32'b01010101010101010101010101010100 && Cout != 1)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0110) begin
                    if(D != 32'b01010101010101010101010101010101 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b0111) begin
```

```
                    if(D != 32'b01010101010101010101010101010101 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1000) begin
                    if(D != 32'b01010101010101010101010101010101 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1001) begin
                    if(D != 32'b00000000000000000000000000000000 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1010) begin
                    if(D != 32'b11111111111111111111111111111111 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1011) begin
                    if(D != 32'b11111111111111111111111111111111 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1100) begin
                    if(D != 32'b11111111111111111111111111111111 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1101) begin
                    if(D != 32'b00000000000000000000000000000000 && Cout != 1)
begin
                        allCorrect = 0;
```

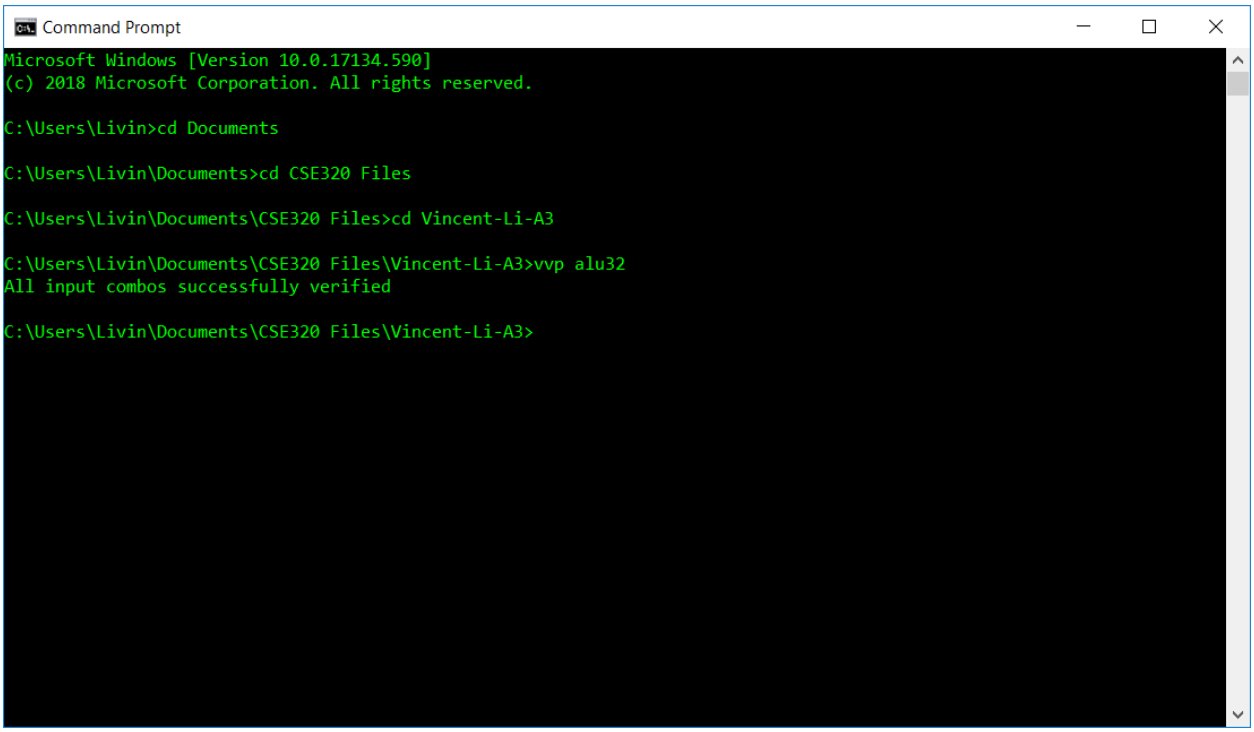
```
                        $display(S);
                    end
                end
                else if(S == 4'b1110) begin
                    if(D != 32'b00000000000000000000000000000001 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end
                else if(S == 4'b1111) begin
                    if(D != 32'b00000000000000000000000000000000 && Cout != 0)
begin
                        allCorrect = 0;
                        $display(S);
                    end
                end

            end
        /*
            S = 4'b0010;
            #1 $display("S = %b\nA = %b\nB = %b\nCin = %b\nD = %b\tCout =
%b\n",S, A, B, Cin, D, Cout);
        */
            if(allCorrect == 1) $display("All input combos successfully
verified");

            $finish;
        end

endmodule
```

Results:




2)

States:

00: waiting for go. Set X and Y from Xi and Yi. Goes to 01

01: Set go to 0. If X and Y are equal, set Xsel and Ysel, and go to 11. Else go to 10

10: Update X and Y based on Xsel and Ysel. Goes to 01

11: Found output

| Current state | PX | PY | go | Xsel | Ysel | NX | NY | xyGCD | Next state |
|---|---|---|---|---|---|---|---|---|---|
| 00 (start) | x | x | 0 | x | x | x | x | x | 00 (start) |
| 00 (start) | x | x | 1 | 0 | 0 | x | x | x | 01 (check) |
| | | | | | | | | | |
| 01 (check) | == | == | 0 | 0 | 0 | x | x | x | 11 (out) |
| | | | | | | | | | |
| 01 (check) | <PY | >PX | 0 | 0 | 1 | PX | PY-PX | x | 10 (comp) |
| 10 (comp) | PX | PY - PX | 0 | 0 | 1 | x | x | x | 01 (check) |
| | | | | | | | | | |
| 01 (check) | >PY | <PX | 0 | 1 | 0 | PX-PY | PY | x | 10 (comp) |
| 10 (comp) | PX-PY | PY | 0 | 1 | 0 | x | x | x | 01 (check) |
| | | | | | | | | | |
| 11 (out) | == | == | 0 | x | x | x | x | PX | 00 (start) |

Code:

```
// Filename: gcd.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
/*
    GCD designed in class, but uses an ALU for all calculations.
*/
module gcd (
    input clk,
    input go,
    input [31 : 0] Xi, Yi,
    output reg [31 : 0] XYGCD
    );
    reg [31 : 0] PX, NX, PY, NY;   // main inputs, present and next
    reg [31 : 0] NGCD; // next output
    reg Xsel, Ysel; // controls to switch between x & x - y and y & y - x
    reg Xld, Yld, Dld;  // pulsed by clock

    // State info
    parameter S0 = 2'b00;  // waiting for go. Set controls and NX and NY from Xi
and Yi. Goes to 01
    parameter S1 = 2'b01;  // Set Xsel and Ysel. If PX and PY are equal, go to
11, else go to 10
    parameter S2 = 2'b10;  // Update NX and NY based on Xsel and Ysel, and go to
01
    parameter S3 = 2'b11;  // Found output
    reg [2 : 0] PS; // present state
    reg [2 : 0] NS = S0; // next state

    // ALU
    reg [3 : 0] S = 4'b0000; // ALU control
    reg Cin = 0;
    wire [31 : 0] D; // ALU output
    wire Cout;
    parameter integer WIDTH = 32;
    alu32 #(.WIDTH(WIDTH)) ALU (.A(PX), .B(PY), .S(S), .Cin(Cin), .D(D),
.Cout(Cout));

    // next state logic
    always @(*) begin
        case(PS)
            S0: begin
                NX = Xi;
```

```
            NY = Yi;
            Xld = 1'b0;
            Yld = 1'b0;
            Dld = 1'b0;
            //Xsel = 0;
            //Ysel = 0;
            if(go == 1) begin
                NS = S1;
            end
            else begin
                NS = S0;
            end
        end

        S1: begin
            S = 4'b0010; // to calculate < or >
            if(D == 0) begin
                Xsel = 0;
                Ysel = 0;
                NS = S3;
            end
            else if(Cout == 0) begin // X > Y
                Xsel = 1;
                Ysel = 0;
                NS = S2;
            end
            else if(Cout == 1) begin // X < Y
                Xsel = 0;
                Ysel = 1;
                NS = S2;
            end
            S = 4'b0000;
        end

        S2: begin
            if(Xsel == 1'b0) begin
                NX = PX;
            end
            else if(Xsel == 1'b1) begin
                NX = PX - PY;
            end
            if(Ysel == 1'b0) begin
                NY = PY;
            end
            else if(Ysel == 1'b1) begin
```

```
                    NY = PY - PX;
                end
                NS = S1;
            end

            S3: begin
                NGCD = PX;
                NS = S0;
            end

        endcase
    end

    // state update
    always @(posedge clk) begin
        //$display("PX = %b\nPY = %b\nXsel = %b\nYsel = %b\nXYGCD = %b\n", PX,
PY, Xsel, Ysel, XYGCD);
        PS <= NS;
        Xld <= ~Xld;
        Yld <= ~Yld;
        Dld <= ~Dld;
        #1;
    end

    // output logic
    always @(posedge clk) begin
        if(Xld == 1'b1) begin
            PX <= NX;
        end
        if(Yld == 1'b1) begin
            PY <= NY;
        end
        if(Dld == 1'b1) begin
            XYGCD <= NGCD;
        end
        #1;
    end

endmodule
```

```
// Filename: gcd_tb.v
// Name: Vincent Li
// Course: CSE320 T/TH 4:30PM
// Professor: Vrudhula
```

```
/*
    Test bench for gcd.v
*/

module gcd_tb;
    reg clk;
    reg go;
    reg [31 : 0] Xi, Yi;
    wire [31 : 0] XYGCD;
    integer i;
    integer allCorrect = 1;

    gcd M(.clk(clk), .go(go), .Xi(Xi), .Yi(Yi), .XYGCD(XYGCD));

    initial
        begin
            go = 0;
            #1;

            // test X > Y
            Xi = 32'b1010;
            Yi = 32'b10;
            #1;
            go = 1;
            #1;
            clk = 0;
            for(i = 0; i < 20; i = i + 1) begin
                clk = ~clk;
                #1;
                clk = ~clk;
                #1;
            end
            if(XYGCD != 32'b1010) begin
                allCorrect = 0;
            end

            // test X < Y
            go = 0;
            #1;
            Xi = 32'b10;
            Yi = 32'b1010;
            #1;
            go = 1;
            #1;
            clk = 0;
```

```
            for(i = 0; i < 20; i = i + 1) begin
                clk = ~clk;
                #1;
                clk = ~clk;
                #1;
            end
            if(XYGCD != 32'b1010) begin
                allCorrect = 0;
            end

            // test X == Y
            go = 0;
            #1;
            Xi = 32'b10;
            Yi = 32'b10;
            #1;
            go = 1;
            #1;
            clk = 0;
            for(i = 0; i < 20; i = i + 1) begin
                clk = ~clk;
                #1;
                clk = ~clk;
                #1;
            end
            if(XYGCD != 32'b10) begin
                allCorrect = 0;
            end


            if(allCorrect == 1) $display("All input combos verified");
            else $display("Not all input combos verified");

            $finish;

        end

endmodule
```

Results:


Command Prompt
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>iverilog -o gcd gcd_tb.v gcd.v alu32.v
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>vvp gcd
Not all input combos verified
Not all input combos verified
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>iverilog -o gcd gcd_tb.v gcd.v alu32.v
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>vvp gcd
Not all input combos verified
Not all input combos verified
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>iverilog -o gcd gcd_tb.v gcd.v alu32.v
gcd.v:56: syntax error
gcd.v:56: error: Malformed conditional expression.
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>iverilog -o gcd gcd_tb.v gcd.v alu32.v
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>vvp gcd
All input combos verified
All input combos verified
C:\Users\Livin\Documents\CSE320 Files\Vincent-Li-A3>