# GROUP PROJECT ON VERILOG

**Group Members:**

1. **Gathik Jindal (089)**
2. **Archit Jaju (128)**
3. **Shubhranil Basak (510)**
4. **Ahilnandan kabilan (614)**

**Q1)**

**RLT CODE:**

```
module one_bit_adder(
   // initalising all the inputs and outputs
   input a, b, c_in,
   output sum, c_out
);
   wire a1 , b1, c_in1, w1, w2, w3, w4, w5, w6, w7;
   // finding sum
   not (a1, a);
   not (b1, b);
   not (c_in1, c_in);
   and (w1, a, b, c_in);     // a.b.c_in
   and (w2, a1, b, c_in1);   // a'.b.c_in'
   and (w3, a1, b1, c_in);   // a'.b'.c_in
   and (w4, a, b1, c_in1);   // a.b'.c_in'
   or (sum, w1, w2, w3, w4); // adding all above
   // finding c_out
   and (w5, a, b);          // a.b
   and (w6, c_in, b);         // b.c_in
   and (w7, a, c_in);         // a.c_in
   or (c_out, w5, w6, w7);  // adding all above
```

```
endmodule;
```

**TEST BENCH:**

```
`include "first_question.v"

module tb_one_bit_adder;

   // Inputs

   reg a, b, c_in;

   // Outputs

   wire sum, c_out;

   // Initiating the one-bit adder module

   one_bit_adder uut ( // uut stands for unit under test

       .a(a),

       .b(b),

       .c_in(c_in),

       .sum(sum),

       .c_out(c_out)

   );

   initial begin

       $monitor("Time=%0t a=%b b=%b c_in=%b sum=%b c_out=%b", $time, a, b, c_in, sum, c_out);

       $dumpfile("quesiton1.vcd");

       $dumpvars();

       a = 0; b = 0; c_in = 0;

       #10;

       a = 1; b = 0; c_in = 1;

       #10;

       a = 1; b = 1; c_in = 0;

       #10;

       a = 0; b = 1; c_in = 1;

       #10;

       a = 1; b = 1; c_in = 1;
```

```
    #10;
    a = 1; b = 0; c_in = 0;
    #10;
    a = 0; b = 1; c_in = 0;
    #10;
    a = 0; b = 0; c_in = 1;
    #10;
    $finish;
  end
Endmodule
```

**OUTPUT:**

```
[Running] tb_first_question.v
VCD info: dumpfile quesiton1.vcd opened for output.
Time=0 a=0 b=0 c_in=0 sum=0 c_out=0
Time=10 a=1 b=0 c_in=1 sum=0 c_out=1
Time=20 a=1 b=1 c_in=0 sum=0 c_out=1
Time=30 a=0 b=1 c_in=1 sum=0 c_out=1
Time=40 a=1 b=1 c_in=1 sum=1 c_out=1
Time=50 a=1 b=0 c_in=0 sum=1 c_out=0
Time=60 a=0 b=1 c_in=0 sum=1 c_out=0
Time=70 a=0 b=0 c_in=1 sum=1 c_out=0
tb_first_question.v:49: $finish called at 80 (1s)
[Done] exit with code=0 in 0.16 seconds
```

**Q2)**

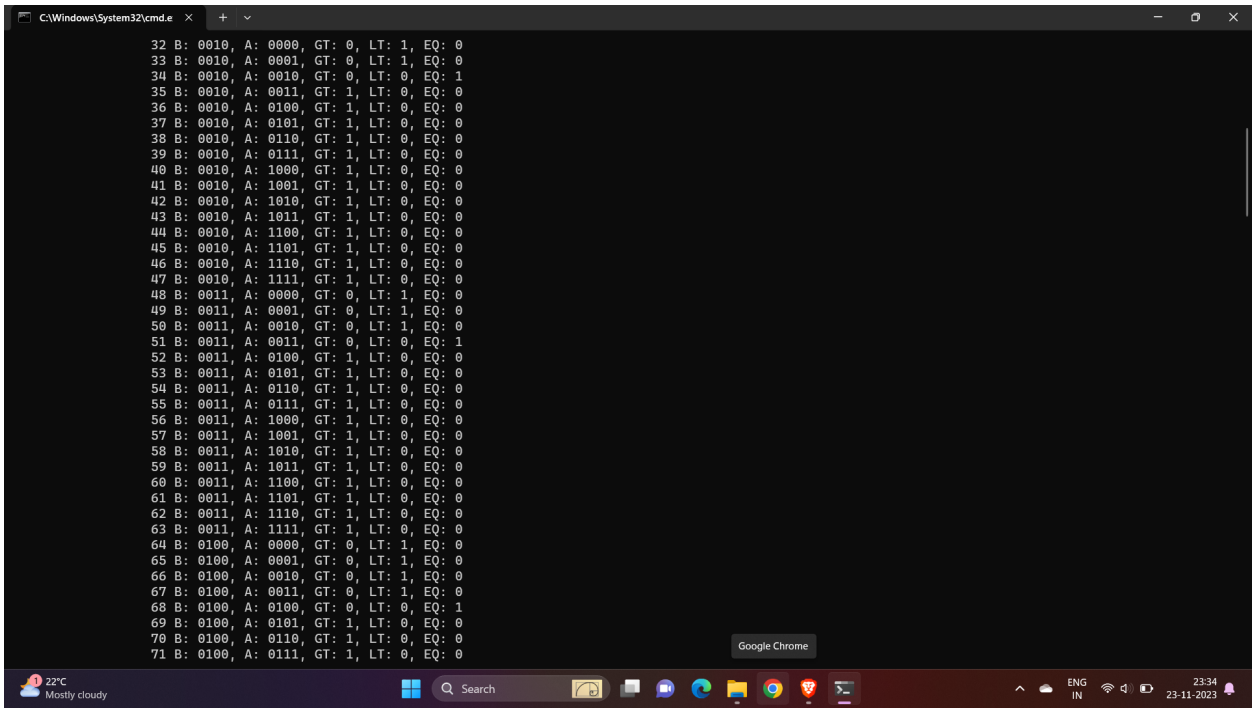
**RTL CODE:**

```
module stimulus();
    reg [3:0] A, B;
    wire A_eq_B, A_gt_B, A_lt_B;
    magnitude_comparator mag(A,B,A_gt_B,A_lt_B,A_eq_B);
    initial begin
        $dumpfile("hope.vcd");
        $dumpvars(0, stimulus);
        A = 4'b0000;
        B = 4'b0000;
```

```
        #255 $finish;
    end
    always
        #1 A[0] = ~A[0];
    always
        #2 A[1] = ~A[1];
    always
        #4 A[2] = ~A[2];
    always
        #8 A[3] = ~A[3];
    always
        #16 B[0] = ~B[0];
    always
        #32 B[1] = ~B[1];
    always
        #64 B[2] = ~B[2];
    always
        #128 B[3] = ~B[3];
    initial
    $monitor("%d B: %b, A: %b, GT: %b, LT: %b, EQ: %b",$time, B, A, A_gt_B,
A_lt_B, A_eq_B);
endmodule
module magnitude_comparator (
    A, B, A_gt_B, A_lt_B, A_eq_B
);
input [3:0] A;
input [3:0] B;
output A_gt_B, A_lt_B, A_eq_B;
wire notA0, notA1, notA2, notA3;
```

```
wire notB0, notB1, notB2, notB3;

wire xnor0, xnor1, xnor2, xnor3;

not (notA0, A[0]);

not (notA1, A[1]);

not (notA2, A[2]);

not (notA3, A[3]);

not (notB0, B[0]);

not (notB1, B[1]);

not (notB2, B[2]);

not (notB3, B[3]);

xnor(xnor3, A[3], B[3]);

xnor(xnor2, A[2], B[2]);

xnor(xnor1, A[1], B[1]);

xnor(xnor0, A[0], B[0]);

wire main1, main2, main3, main4;

and(main1, notB3, A[3]);

and(main2, xnor3, A[2], notB2);

and(main3, xnor3, xnor2, A[1], notB1);

and(main4, xnor3, xnor2, xnor1, A[0], notB0);

or(A_gt_B, main1, main2, main3, main4);

wire main11, main12, main13, main14;

and(main11, notA3, B[3]);

and(main12, xnor3, B[2], notA2);

and(main13, xnor3, xnor2, B[1], notA1);

and(main14, xnor3, xnor2, xnor1, B[0], notA0);

or(A_lt_B, main11, main12, main13, main14);

and(A_eq_B, xnor0, xnor1, xnor2, xnor3);

endmodule
```

**OUTPUT:**




**Q3)**

**RTL CODE:**

```
// Code your design here

module mux_2to1 (

    input a,

    input b,

    input sel,

    output reg out

);

    always @(*)

    begin

        case(sel)

            1'b0: out = a;

            1'b1: out = b;

            default: out = 1'bx;

        endcase

    end
```

```
endmodule

module or1(

    input a,

    input b,

    output reg out

);



    always @(*) begin

        // Behavioral logic for OR gate

        out = a | b; // OR operation using bitwise OR

    end

endmodule

module question_3(

    input i1,

    input i2,

    input i3,

    input i4,

    input i5,

    input i6,

    input i7,

    input i8,

    output f

);

    wire x1, x2, x3;

    mux_2to1 mux1(

        .a(i4),

        .b(i5),

        .sel(i3),

        .out(x1)
```

```
    );
    mux_2to1 mux2(
        .a(i7),
        .b(i8),
        .sel(i6),
        .out(x2)
    );

    // OR gate instantiation
    or1 a1(
        .out(x3),
        .a(i1),
        .b(i2)
    );
    mux_2to1 mux3(
        .a(x1),
        .b(x2),
        .sel(x3),
        .out(f) // Missing comma corrected here
    );
endmodule
module stimulus;
    reg i1, i2, i3, i4, i5, i6, i7, i8;
    wire f;
    question_3 uut (
        .i1(i1),
        .i2(i2),
        .i3(i3),
        .i4(i4),
```

```
        .i5(i5),
        .i6(i6),
        .i7(i7),
        .i8(i8),
        .f(f)
    );
    initial begin
        i2=0;
        i3=0;
        i5=0;
        i6=0;
        i7=0;
        i4 = 0;
        i1 = 0;
        i8 = 1;
        // Add more initial statements if needed for your test scenario
        // Your testbench actions, stimulus, and checks go here
        // For example, changing input values over time
        $monitor("w1=%b w2=%b w3=%b w4=%b f=%b",i2,i3,i5,i7,f);
        #10 i2=0;i3=0;i6=0;i7=0;i5=0;
        #10 i2=1;i3=1;i6=1;i7=1;i5=1;
        #10 i2=0;i3=0;i6=0;i7=0;i6=1;
    end
endmodule
```

**OUTPUT:**

```
[Running] third_questioin.v
w1=0 w2=0 w3=0 w4=0 f=0
w1=1 w2=1 w3=1 w4=1 f=1
w1=0 w2=0 w3=1 w4=0 f=0
[Done] exit with code=0 in 0.192 seconds
```

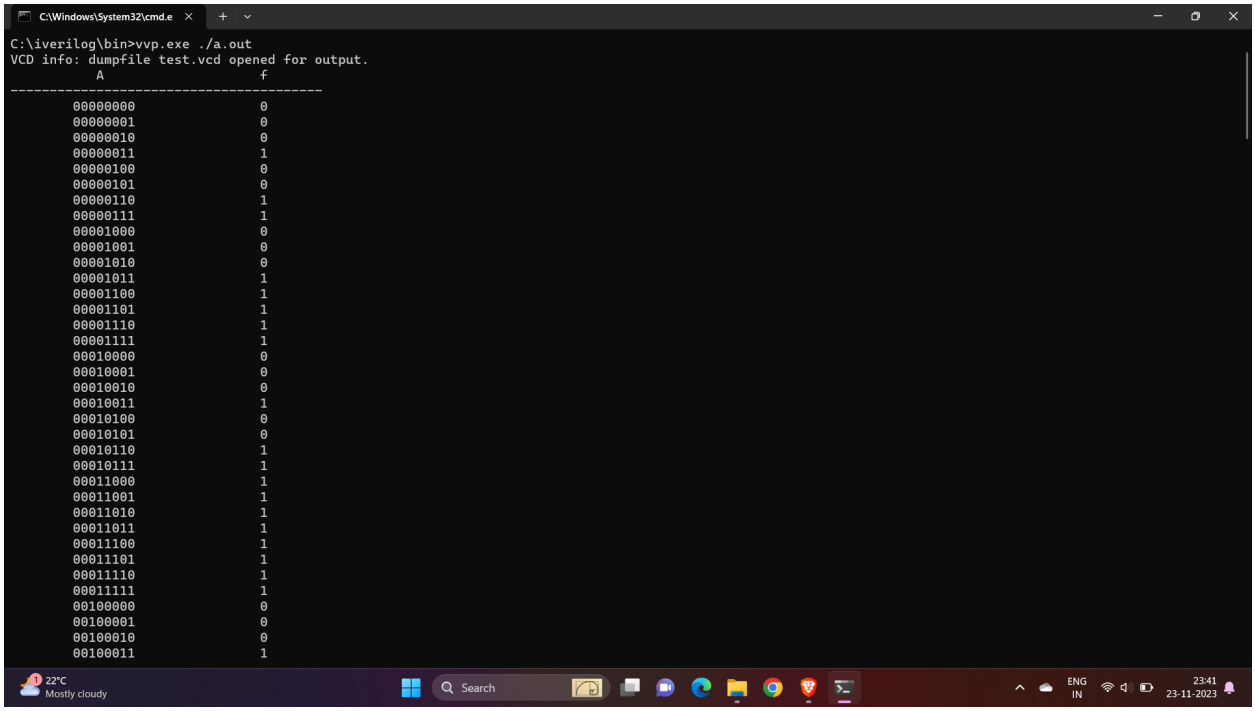
**Q4)**

**RTL CODE:**

```
// when the following blocking statements are converted to non blocking
statements

// unexpected behavior is taking place and we cannot really predict the output

// an observation to be noted is that if only the second blocking statement is
made

// a blocking statement then only the start and the ending ones are detected

// rest all are ignored

module stimulus();

    reg [7:0] A;

    wire f;

    AdjacentOnes ad(A, f);

    initial begin

        $dumpfile("test.vcd");

        $dumpvars(0, stimulus);

        A = 8'b00000000;

        #255 $finish;

    end

    always

        #1  A[0] = ~A[0];

    always

        #2  A[1] = ~A[1];

    always

        #4  A[2] = ~A[2];

    always

        #8  A[3] = ~A[3];

    always

        #16  A[4] = ~A[4];

    always

        #32  A[5] = ~A[5];

    always
```

```
        #64  A[6] = ~A[6];
    always
        #128  A[7] = ~A[7];
    initial
    $display("\t   A\t\t\tf");
    initial
    $display("----------------------------------------");
    initial
    $monitor("\t%b\t\t%d", A, f);
endmodule
module AdjacentOnes (
    input [7:0] A,  // 8-bit binary vector input
    output reg f     // Output
);
    integer k;
    always @ (A)
    begin
        f = A[1] & A[0];
        for (k = 2; k < 8; k = k + 1)
            f = f | (A[k] & A[k - 1]);
    end
endmodule
```

## OUTPUT:


C:\Windows\System32\cmd.e
C:\iverilog\bin>vvp.exe ./a.out
VCD info: dumpfile test.vcd opened for output.
A f
00000000 0
00000001 0
00000010 0
00000011 1
00000100 0
00000101 0
00000110 1
00000111 1
00001000 0
00001001 0
00001010 0
00001011 1
00001100 1
00001101 1
00001110 1
00001111 1
00010000 0
00010001 0
00010010 0
00010011 1
00010100 0
00010101 0
00010110 1
00010111 1
00011000 1
00011001 1
00011010 1
00011011 1
00011100 1
00011101 1
00011110 1
00011111 1
00100000 0
00100001 0
00100010 0
00100011 1
22°C
Mostly cloudy
Search
ENG
IN
23:41
23-11-2023