Digital IC Design
Assignment 2: Vectoring Mode CORDIC

Name- Pushkal Mishra
Roll- EE20BTECH11042

Design of a 2D Vectoring mode Circular CORDIC with 8 bit word length.

Implementation

```
module cordic_vectoring_stage (x_out, y_out, output_angle, clock, reset, x_in, y_in,
input_angle, n_shifts);

    output reg [31:0] x_out, y_out, output_angle;

    input clock, reset;
    input [31:0] x_in, y_in, input_angle;
    input [2:0] n_shifts;

    reg [31:0] shift_angle;

    initial
    begin

        x_out <= 32'd00_00;
        y_out <= 32'd00_00;
        output_angle <= 32'd00_00;

        case (n_shifts)

            3'b000: shift_angle <= 32'd45_000000;
            3'b001: shift_angle <= 32'd26_565051;
            3'b010: shift_angle <= 32'd14_036243;
            3'b011: shift_angle <= 32'd07_125016;
            3'b100: shift_angle <= 32'd03_576334;
            3'b101: shift_angle <= 32'd01_789910;
            3'b110: shift_angle <= 32'd00_895173;
            3'b111: shift_angle <= 32'd00_447614;

        endcase

    end

    always @(posedge clock, reset)
    begin

        if (reset)
        begin
```

```
            x_out <= 32'd00_00;
            y_out <= 32'd00_00;
            output_angle <= 32'd00_00;
        end

        else
        begin

            if (y_in >= 0)
            begin

                x_out <= x_in + (y_in >> n_shifts);
                y_out <= y_in - (x_in >> n_shifts);
                output_angle = input_angle + shift_angle;

            end

            else
            begin
                x_out <= x_in - (y_in >> n_shifts);
                y_out <= y_in + (x_in >> n_shifts);
                output_angle = input_angle - shift_angle;
            end

        end

    end

endmodule

module CORDIC_Vectoring (norm, angle, clock, reset, input_x, input_y);

    output reg [31:0] norm, angle;

    input clock, reset;
    input [31:0] input_x, input_y;

    wire [31:0] x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8;
    wire [31:0] y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8;
    wire [31:0] angle_1, angle_2, angle_3, angle_4, angle_5, angle_6, angle_7, angle_8;

    cordic_vectoring_stage stage1 (x_1, y_1, angle_1, clock, reset, input_x, input_y,
32'd00_00, 3'b000);
    cordic_vectoring_stage stage2 (x_2, y_2, angle_2, clock, reset, x_1, y_1, angle_1,
3'b001);
    cordic_vectoring_stage stage3 (x_3, y_3, angle_3, clock, reset, x_2, y_2, angle_2,
3'b010);
    cordic_vectoring_stage stage4 (x_4, y_4, angle_4, clock, reset, x_3, y_3, angle_3,
3'b011);
    cordic_vectoring_stage stage5 (x_5, y_5, angle_5, clock, reset, x_4, y_4, angle_4,
```

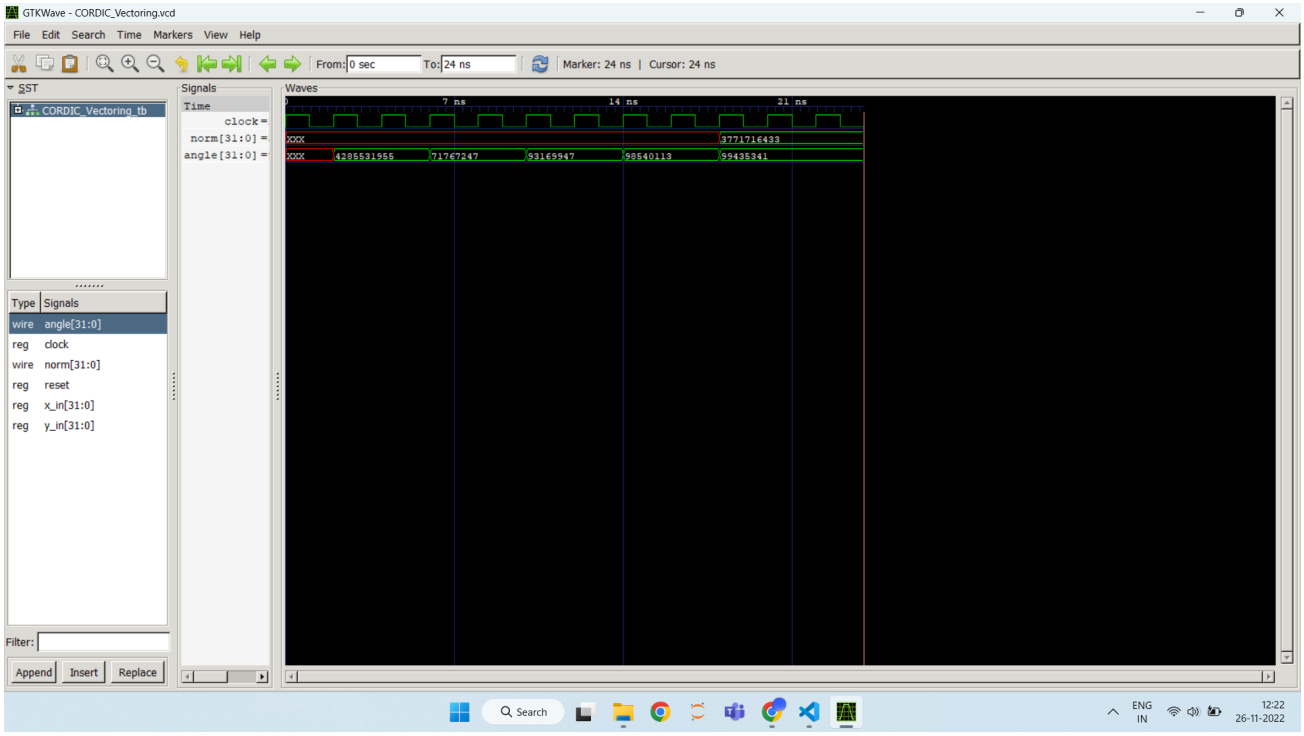
```
3'b100);
    cordic_vectoring_stage stage6 (x_6, y_6, angle_6, clock, reset, x_5, y_5, angle_5,
3'b101);
    cordic_vectoring_stage stage7 (x_7, y_7, angle_7, clock, reset, x_6, y_6, angle_6,
3'b110);
    cordic_vectoring_stage stage8 (x_8, y_8, angle_8, clock, reset, x_7, y_7, angle_7,
3'b111);

    reg [31:0] final_cos_product = 32'd00_60725911;

    initial
    begin

        norm <= 32'd00_00;
        angle <= 32'd00_00;

    end

    always @(posedge clock, reset)
    begin

        if (reset)
        begin

            norm <= 32'd00_00;
            angle <= 32'd00_00;

        end

        else
        begin

            norm <= x_8 * final_cos_product;
            angle <= angle_8;

        end
    end

endmodule
```

Note that there are 8 stages, i.e. the CORDIC performs 8 coordinate rotations.

Testbench

```
`timescale 1ns/1ns
`include "CORDIC_Vectoring.v"

module CORDIC_Vectoring_tb();
    wire [31:0] norm, angle;
    reg [31:0] x_in, y_in;
    reg clock = 1'b1;
    reg reset = 1'b0;
    always #1 clock = ~clock;

    initial
    begin
        x_in <= 32'd00_00;
        y_in <= 32'd10_00;
    end

    CORDIC_Vectoring mod (norm, angle, clock, reset, x_in, y_in);

    initial
    begin
        $dumpfile("CORDIC_Vectoring.vcd");
        $dumpvars;
        #24 $finish;
    end

endmodule
```

Output in GTKWave



Digital IC Design
Assignment 2: Rotation Mode CORDIC

Name- Pushkal Mishra
Roll- EE20BTECH11042

Design of a 2D Rotation mode Circular CORDIC with 8 bit word length.

Implementation

```
module cordic_rotation_stage (x_out, y_out, output_angle, clock, reset, input_x, input_y,
current_angle, rotation_angle, n_shifts);

    output reg [31:0] x_out, y_out, output_angle;
    input clock, reset;
    input [31:0] input_x, input_y, current_angle, rotation_angle;
    input [2:0] n_shifts;
    reg [31:0] shift_angle;

    initial
    begin
        x_out <= 32'd00_00;
        y_out <= 32'd00_00;
        output_angle <= 32'd00_00;
        case (n_shifts)
            3'b000: shift_angle <= 32'd45_000000;
            3'b001: shift_angle <= 32'd26_565051;
            3'b010: shift_angle <= 32'd14_036243;
            3'b011: shift_angle <= 32'd07_125016;
            3'b100: shift_angle <= 32'd03_576334;
            3'b101: shift_angle <= 32'd01_789910;
            3'b110: shift_angle <= 32'd00_895173;
            3'b111: shift_angle <= 32'd00_447614;
        endcase
    end

    always @(posedge clock, reset)
    begin
        if (reset)
        begin
            x_out <= 32'd00_00;
            y_out <= 32'd00_00;
            output_angle <= current_angle;
        end

        else
        begin
            if (current_angle <= rotation_angle)
```

```
            begin
                x_out <= input_x + (input_y >> n_shifts);
                y_out <= input_y - (input_x >> n_shifts);
                output_angle = current_angle + shift_angle;
            end

            else
            begin
                x_out <= input_x - (input_y >> n_shifts);
                y_out <= input_y + (input_x >> n_shifts);
                output_angle = current_angle - shift_angle;
            end
        end
    end
endmodule
```

```
module CORDIC_Rotation (x_out, y_out, clock, reset, input_x, input_y, rotation_angle);

    output reg [31:0] x_out, y_out;
    input clock, reset;
    input [31:0] input_x, input_y, rotation_angle;
    wire [31:0] x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8;
    wire [31:0] y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8;
    wire [31:0] angle_1, angle_2, angle_3, angle_4, angle_5, angle_6, angle_7, angle_8;

    cordic_rotation_stage stage1 (x_1, y_1, angle_1, clock, reset, input_x, input_y,
32'd00_00, rotation_angle, 3'b000);
    cordic_rotation_stage stage2 (x_2, y_2, angle_2, clock, reset, x_1, y_1, angle_1,
rotation_angle, 3'b001);
    cordic_rotation_stage stage3 (x_3, y_3, angle_3, clock, reset, x_2, y_2, angle_2,
rotation_angle, 3'b010);
    cordic_rotation_stage stage4 (x_4, y_4, angle_4, clock, reset, x_3, y_3, angle_3,
rotation_angle, 3'b011);
    cordic_rotation_stage stage5 (x_5, y_5, angle_5, clock, reset, x_4, y_4, angle_4,
rotation_angle, 3'b100);
    cordic_rotation_stage stage6 (x_6, y_6, angle_6, clock, reset, x_5, y_5, angle_5,
rotation_angle, 3'b101);
    cordic_rotation_stage stage7 (x_7, y_7, angle_7, clock, reset, x_6, y_6, angle_6,
rotation_angle, 3'b110);
    cordic_rotation_stage stage8 (x_8, y_8, angle_8, clock, reset, x_7, y_7, angle_7,
rotation_angle, 3'b111);

    reg [31:0] final_cos_product = 32'd00_60725911;
    always @(posedge clock, reset)
    begin
        if (reset)
        begin
            x_out <= 32'd00_00;
            y_out <= 32'd00_00;
```

```
        end

        else
        begin
            x_out <= (x_8 * final_cos_product);
            y_out <= (y_8 * final_cos_product);
        end
    end
endmodule
```

Testbench

```
`timescale 1ns/1ns
`include "CORDIC_Rotation.v"

module CORDIC_Rotation_tb ();

    wire [31:0] x_out, y_out;
    reg [31:0] x_in, y_in, rotation_angle;

    reg clock = 1'b0;
    reg reset = 1'b0;

    always #1 clock = ~clock;

    initial
    begin

        x_in <= 32'd0;
        y_in <= 32'd1_0;
        rotation_angle <= 32'd90_0;

    end
```

```
     CORDIC_Rotation mod (x_out, y_out, clock, reset, x_in, y_in, rotation_angle);

    initial
    begin
        $dumpfile("CORDIC_Rotation.vcd");
        $dumpvars;
        #20 $finish;
     end
endmodule
```

Output in GTKWave