

```
`timescale 1ns/1ps
```

```
module vectoring_cordic
```

```
(
```

```
input clk,
```

```
input signed[15:0]x_in,
```

```
input signed[15:0]y_in,
```

```
output wire signed[19:0] x_final,final_angle
```

```
);
```

```
reg signed [19:0]angle[0:19]; //angle
```

```
reg signed [19:0]s[0:19]; //sign
```

```
reg signed [19:0]x[0:19]; //x
```

```
reg signed [19:0]y[0:19]; //y
```

```
// Declaring Output of Intermediate Stages
```

```
wire signed[19:0] out0,out0_1;
```

```
wire signed[19:0] out1,out1_1;
```

```
wire signed[19:0] out2,out2_1;
```

```
wire signed[19:0] out3,out3_1;
```

```
wire signed[19:0] out4,out4_1;
```

```
wire signed[19:0] out5,out5_1;
```

```
wire signed[19:0] out6,out6_1;
```

```
wire signed[19:0] out7,out7_1;
```

```
//Declaring Microrotation (All the angles are left shifted by 8 bit)
```

```
wire signed [19:0]micro_rotation[0:7];
```

```
assign micro_rotation[0]=19'd11520;
```

```
assign micro_rotation[1]=19'd6801;
```

```
assign micro_rotation[2]=19'd3593;
```

```
assign micro_rotation[3]=19'd1824;
```

```
assign micro_rotation[4]=19'd916;
```

```

assign micro_rotation[5]=19'd458;
assign micro_rotation[6]=19'd229;
assign micro_rotation[7]=19'd115;

```

```

wire signed [19:0]d,shift,final;

```

```

// Instantiating Adder modules

```

```

Adder d0 (x_in*256,(y_in*256)<0 ? -y_in*256 : y_in*256, out0);
Adder d1 (y_in*256,(y_in*256)<0 ? x_in*256 :-x_in*256, out0_1);
Adder d2 (x[0],(y[0]<0 ? -(y[0]>>>1) : y[0]>>>1, out1);
Adder d3 (y[0],(y[0]<0 ? x[0]>>>1 : -(x[0]>>>1), out1_1);
Adder d4 (x[1],(y[1]<0 ? -(y[1]>>>2) : y[1]>>>2, out2);
Adder d5 (y[1],(y[1]<0 ? x[1]>>>2 : -(x[1]>>>2), out2_1);
Adder d6 (x[2],(y[2]<0 ? -(y[2]>>>3) : y[2]>>>3, out3);
Adder d7 (y[2],(y[2]<0 ? x[2]>>>3 : -(x[21]>>>3), out3_1);
Adder d8 (x[3],(y[3]<0 ? -(y[3]>>>4) : y[3]>>>4, out4);
Adder d9 (y[3],(y[3]<0 ? x[3]>>>4 : -(x[3]>>>4), out4_1);
Adder d10 (x[4],(y[4]<0 ? -(y[4]>>>5) : y[4]>>>5, out5);
Adder d11 (y[4],(y[4]<0 ? x[4]>>>5 : -(x[4]>>>5), out5_1);
Adder d12 (x[5],(y[5]<0 ? -(y[5]>>>6) : y[5]>>>6, out6);
Adder d13 (y[5],(y[5]<0 ? x[5]>>>6 : -(x[5]>>>6), out6_1);
Adder d14 (x[6],(y[6]<0 ? -(y[6]>>>7) : y[6]>>>7, out7);
Adder d15 (y[6],(y[6]<0 ? x[6]>>>7 : -(x[6]>>>7), out7_1);

```

```

always @(posedge clk)

```

```

begin

```

```

angle[0]<= (y_in*256)<0 ? 0-micro_rotation[0] : 0+micro_rotation[0];

```

```

s[0]<= y_in*256<0 ? 1: -1;

```

```

x[0]<=out0;

```

```

y[0]<=out0_1;

```

```

angle[1]<= (out0_1)<0 ? angle[0]-micro_rotation[1]:

```

angle[0]+micro\_rotation[1];

s[1]<= angle[0]<0 ?1: -1;

x[1]<=out1;

y[1]<=out1\_1;

angle[2]<= (out1\_1)<0 ? angle[1]-micro\_rotation[2]:

angle[1]+micro\_rotation[2];

s[2]<=angle[1]>0 ? 1:-1;

x[2]<=out2;

y[2]<=out2\_1;

angle[3]<=(out2\_1)<0 ? angle[2]-micro\_rotation[3]:angle[2]+micro\_rotation[3];

s[3]<=angle[2]>0 ? 1:-1;

x[3]<=out3;

y[3]<=out3\_1;

angle[4]<=(out3\_1)<0 ? angle[3]-micro\_rotation[4]:angle[3]+micro\_rotation[4];

s[4]<=angle[3]>0 ? 1:-1;

x[4]<=out4;

y[4]<=out4\_1;

angle[5]<=(out4\_1)<0 ? angle[4]-micro\_rotation[5]:angle[4]+micro\_rotation[5];

s[5]<=angle[4]>0 ? 1:-1;

x[5]<=out5;

y[5]<=out5\_1;

angle[6]<=(out5\_1)<0 ? angle[5]-micro\_rotation[6]:angle[5]+micro\_rotation[6];

s[6]<=angle[5]>0 ? 1:-1;

x[6]<=out6;

y[6]<=out6\_1;

```
angle[7]<=(out6_1)<0 ? angle[6]-micro_rotation[7]:angle[6]+micro_rotation[7];
```

```
s[7]<=angle[6]>0 ? 1:-1;
```

```
x[7]<=out7;
```

```
y[7]<=out7_1;
```

```
end
```

```
assign x_final=0.607*out7;
```

```
assign final_angle=angle[7];
```

```
endmodule
```

```
//Adder Module
```

```
    `timescale 1ns/1ps
```

```
    module Adder(
```

```
        input signed[15:0] y1,
```

```
        input signed[15:0] y2,
```

```
        output wire signed[19:0] y
```

```
    );
```

```
        assign y=y1+y2;
```

```
endmodule
```

```
TESTBENCH
```

```
// Code your testbench here
```

```
// or browse Examples
```

```
`timescale 1ns/1ps
```

```
module test_bench;
```

```

//Inputs
reg clk;
reg signed[15:0] y_in;
reg signed[15:0] x_in;

//Outputs
wire signed[19:0] x_final;
wire signed[19:0] final_angle;

//instantiate the device under test
vectoring_cordic
v1(
.clk(clk), .y_in(y_in), .x_in(x_in), .x_final(x_final), .final_angle(final_angle)
);
localparam scale = 2**(-8.0);

initial begin
clk=1;
#500
$display("\t x_in is %d",x_in);
$display("\t y_in is %d", y_in);
$display("\t norm is %f", (x_final*scale));
$display("\t final angle is %f", (final_angle*scale));

end

always #25 clk=~clk;

initial begin
x_in = 15'd10;
y_in = 15'd30;

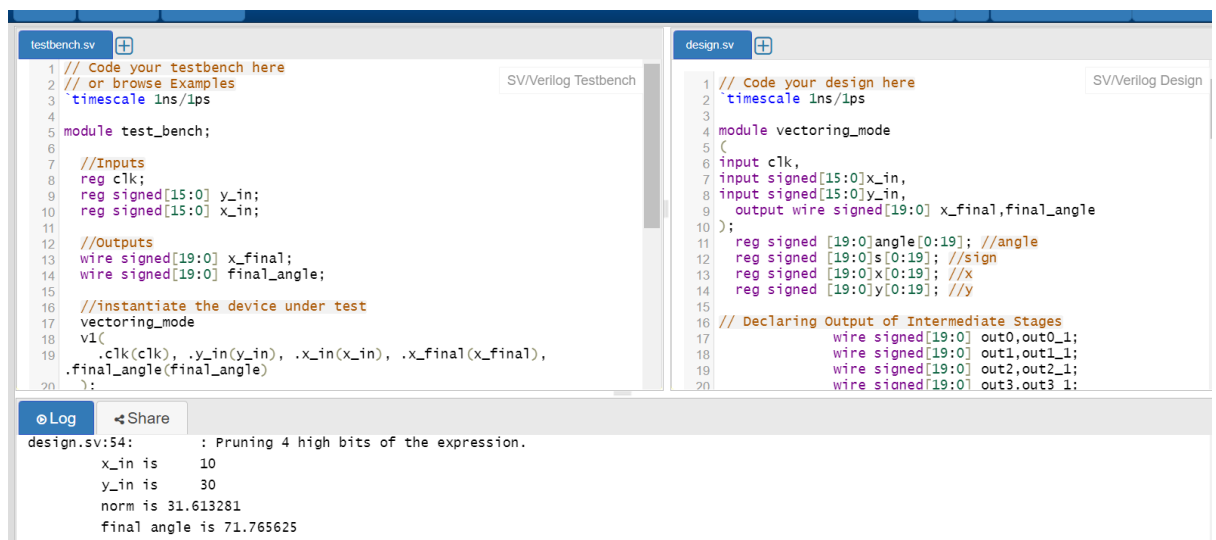
```

end

endmodule

## OUTPUT

```
x_in is      10
y_in is      30
norm is 31.613281
final angle is 71.765625
```



The screenshot displays a Verilog development environment with two source files and a log window.

**testbench.v** (SV/Verilog Testbench):

```
1 // Code your testbench here
2 // or browse Examples
3 timescale 1ns/1ps
4
5 module test_bench;
6
7 //Inputs
8 reg clk;
9 reg signed[15:0] y_in;
10 reg signed[15:0] x_in;
11
12 //Outputs
13 wire signed[19:0] x_final;
14 wire signed[19:0] final_angle;
15
16 //instantiate the device under test
17 vectoring_mode
18 v1(
19     .clk(clk), .y_in(y_in), .x_in(x_in), .x_final(x_final),
20     .final_angle(final_angle)
21 );
```

**design.v** (SV/Verilog Design):

```
1 // Code your design here
2 timescale 1ns/1ps
3
4 module vectoring_mode
5 (
6     input clk,
7     input signed[15:0] x_in,
8     input signed[15:0] y_in,
9     output wire signed[19:0] x_final, final_angle
10 );
11     reg signed [19:0] angle[0:19]; //angle
12     reg signed [19:0] s[0:19]; //sign
13     reg signed [19:0] x[0:19]; //x
14     reg signed [19:0] y[0:19]; //y
15
16 // Declaring Output of Intermediate Stages
17     wire signed[19:0] out0,out0_1;
18     wire signed[19:0] out1,out1_1;
19     wire signed[19:0] out2,out2_1;
20     wire signed[19:0] out3,out3_1;
```

**Log Window:**

```
design.v:54:      : Pruning 4 high bits of the expression.
x_in is      10
y_in is      30
norm is 31.613281
final angle is 71.765625
```

## OUTPUT2

```
x_in is      20
y_in is      45
norm is 49.244289
final angle is 71.765625
```

## OUTPUT3

```
x_in is      10
y_in is      40
norm is 44.721359
final angle is 76.242188
```