DIGITAL IC DESIGN LAB

AMREEN KAUR

IS21MTECH14002

2D-CORDIC VECTORING MODE-CODE,TEST BENCH,SIMULATIONS AND VALIDATIONS

QUESTION STATEMENT-Upload the Code for 2D Vectoring Mode CORDIC along with Test Bench, Validation/ simulation results. Include everything in a PDF File and Upload as a Single file.

CODE FOR CORDIC-VECTORING MODE

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:AMREEN KAUR
//
// Create Date: 23.04.2022 17:26:27
// Design Name:CORDIC VECTORING MODE
// Module Name: angletable
// Project Name:CORDIC
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vectoring();
    reg [15:0] m [0:8];
    reg signed [15:0] mf [0:7];
    reg clk;
    reg signed [32:0] x1 [0:8];
    reg signed [32:0] y1 [0:8];
    reg signed [32:0] y2;
    reg signed [32:0] x2;


    initial begin:my_fn
        integer i;

        m[0] =16'b0001_0001_1001_0100;
        m[1] =16'b0000_1010_0000_0111;
        m[2] =16'b0000_0101_0111_1011;
        m[3] =16'b0000_0010_1100_1000;
```

→ code info

→ registers declared for respective i/ps & o/ps.

```
    m[4] =16'b0000_0001_0110_0101;
    m[5] =16'b0000_0000_1011_0011;
    m[6] =16'b0000_0000_0101_1001;
    m[7] =16'b0000_0000_0010_1100;
    m[8] =16'b0000_0000_0001_0110;
```

← arctan table with properly modified L's

```
    //angle = 16'b0001110010110110;
end
```

↙ logic for computations

```
// if ( m[0] > angle)begin
//outputc [0]=(m[0]> angle) ?(m[0]+m[1]):m[0]-m[1]);
//if(m[0]+m[1])
//{
//}

/*@(posedge clk)
if ( m[0] > angle) begin
    mf[0] = m[0]-m[1];
    outputc[0] = 8'b00000000;end

else if ( m[0] < angle)begin
    mf[0]= m[0]+m[1];
    outputc[0] = 8'b00000001;end

@(posedge clk)
if ( mf[0] > angle) begin
    mf[1] = mf[0]-m[2];outputc[1] = 8'b00000000;
end

else if ( mf[0] < angle)begin
    mf[1] = mf[0]+m[2];outputc[1] = 8'b00000001;
end


@(posedge clk)
if ( mf[1] > angle) begin
    mf[2] = mf[1]-m[3];
```

```
    mf[2] = mf[1]-m[3];
    outputc[2] = 8'b00000000;end

else if ( mf[1] < angle)begin
    mf[2] = mf[1]+m[3];
    outputc[2] = 8'b00000001;end

@(posedge clk)
if ( mf[2] > angle) begin
    mf[3] = mf[2]-m[4];
    outputc[3] = 8'b00000000;end

else if ( mf[2] < angle)begin
    mf[3] = mf[2]+m[4];
    outputc[3] = 8'b00000001;end

@(posedge clk)
if ( mf[3] > angle) begin
    mf[4] = mf[3]-m[5];
    outputc[4] = 8'b00000000;end

else if ( mf[3] < angle)begin
    mf[4] = mf[3]+m[5];
    outputc[4] = 8'b00000001;end

@(posedge clk)
if ( mf[4] > angle) begin
    mf[5] = mf[4]-m[6];
    outputc[5] = 8'b00000000;end

else if ( mf[4] < angle)begin
    mf[5] = mf[4]+m[6];
    outputc[5] = 8'b00000001;end

@(posedge clk)
if ( mf[5] > angle) begin
    mf[6] = mf[5]-m[7];
    outputc[6] = 8'b00000000;end
```

```
    else if ( mf[5] < angle)begin
        mf[6] = mf[5]+m[7];
        outputc[6] = 8'b00000001;end

    @(posedge clk)
    if ( mf[6] > angle) begin
        mf[7] = mf[6]-m[8];
        outputc[7] = 8'b00000000;end

    else if ( mf[6] < angle)begin
        mf[7] = mf[6]+m[8];
        outputc[7] = 8'b00000001;end



    $display("[0%d]",outputc[i]);
end:my_fn*/
    initial begin
        clk= 0;
        forever
            #5 clk= ~clk;
    end
    initial
    begin
        x1[0]=32'b101110111000;//3000

        y1[0]=32'b111110100000;//4000
        @(posedge clk)

        x1[1]  = x1[0]+y1[0];
        y1[1]  = -x1[0]+y1[0];
        mf[0]=m[0];

        @(posedge clk)
        if (y1[1] < 0) begin
            y2= y1[1]>>>1;
            x1[2]  = x1[1]-y2;
            x2= x1[1]>>>1;
```

```
    x2= x1[1]>>>1;
    y1[2]  = x2+y1[1];
    mf[1]=mf[0]+m[1];
end
else if(y1[1] > 0) begin
    y2= y1[1]>>>1;
    x1[2]  = x1[1]+y2;
    x2= x1[1]>>>1;
    y1[2]  = -x2+y1[1];
    mf[1]=mf[0]-m[1];
end


@(posedge clk)
if (y1[2] < 0) begin
    y2 = y1[2] >>> 2;
    x1[3]= x1[2]- y2;
    x2= x1[2] >>> 2;
    y1[3]  = x2 + y1[2];
    mf[2]=mf[1]+m[2];
end
else if(y1[2] > 0) begin
    y2= y1[2]>>>2;
    x1[3]= x1[2]+y2;
    x2= x1[2]>>>2;
    y1[3]  = -x2+y1[2];
    mf[2]=mf[1]-m[2];
end

@(posedge clk)
if (y1[3] < 0) begin
    y2= y1[3] >>> 3;
    x1[4]  = x1[3]-y2;
    x2= x1[3]>>>3;
    y1[4]  = x2+y1[3];
    mf[3]=mf[2]+m[3];
end
else if(y1[3] > 0) begin
    y2= y1[3]>>>3;
```

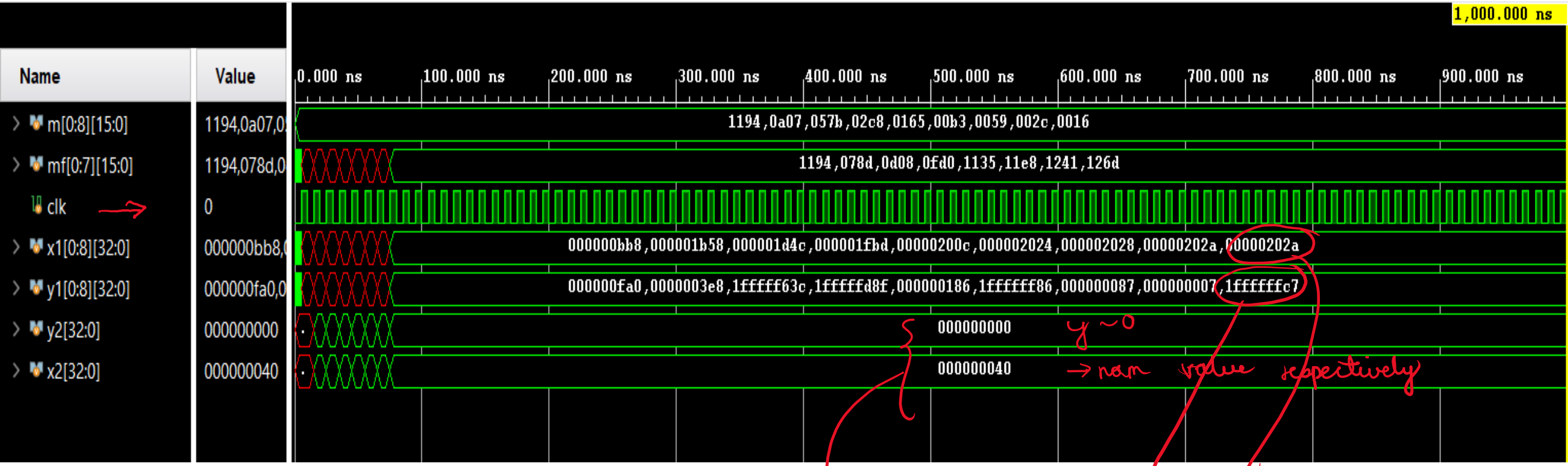
```
    x1[4]  = x1[3]+y2;
    x2= x1[3]>>>3;
    y1[4]  = -x2+y1[3];
    mf[3]=mf[2]+m[3];
end

@(posedge clk)
if (y1[4] < 0) begin
    y2= y1[4]>>>4;
    x1[5]  = x1[4]-y2;
    x2= x1[4]>>>4;
    y1[5]  = x2+y1[4];
    mf[4]=mf[3]+m[4];
end
else if(y1[4] > 0) begin
    y2= y1[4]>>>4;
    x1[5]  = x1[4]+y2;
    x2= x1[4]>>>4;
    y1[5]  = -x2+y1[4];
    mf[4]=mf[3]+m[4];
end

@(posedge clk)
if (y1[5] < 0) begin
    y2= y1[5]>>>5;
    x1[6]  = x1[5]-y2;
    x2= x1[5]>>>5;
    y1[6]  = x2+y1[5];
    mf[5]=mf[4]+m[5];
end
else if(y1[5] > 0) begin
    y2= y1[5]>>>5;
    x1[6]  = x1[5]+y2;
    x2= x1[5]>>>5;
    y1[6]  = -x2+y1[5];
    mf[5]=mf[4]+m[5];
end
```

```
        @(posedge clk)
        if (y1[6] < 0) begin
            y2= y1[6]>>>6;
            x1[7]  = x1[6]-y2;
            x2= x1[6]>>>6;

            y1[7]  = x2+y1[6];
            mf[6]=mf[5]+m[6];

        end
        else if(y1[6] > 0) begin
            y2= y1[6]>>>6;
            x1[7]  = x1[6]+y2;
            x2= x1[6]>>>6;
            y1[7]  = -x2+y1[6];
            mf[6]=mf[5]+m[6];
        end

        @(posedge clk)
        if (y1[7] < 0) begin
            y2= y1[7]>>>7;
            x1[8]  = x1[7]-y2;
            x2= x1[7]>>>7;
            y1[8]  = x2+y1[7];
            mf[7]=mf[6]+m[7];

        end
        else if(y1[7] > 0) begin
            y2= y1[7]>>>7;
            x1[8]  = x1[7]+y2;
            x2= x1[7]>>>7;
            y1[8]  = -x2+y1[7];
            mf[7]=mf[6]+m[7];
        end
    end
endmodule
```

TESTBENCH

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 30.04.2022 22:13:56
// Design Name:
// Module Name: vectoring_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module vectoring_tb;


vectoring va1();



endmodule
```

OUTPUT SIMULATIONS IN HEX



y ~ 0

→ non value respectively

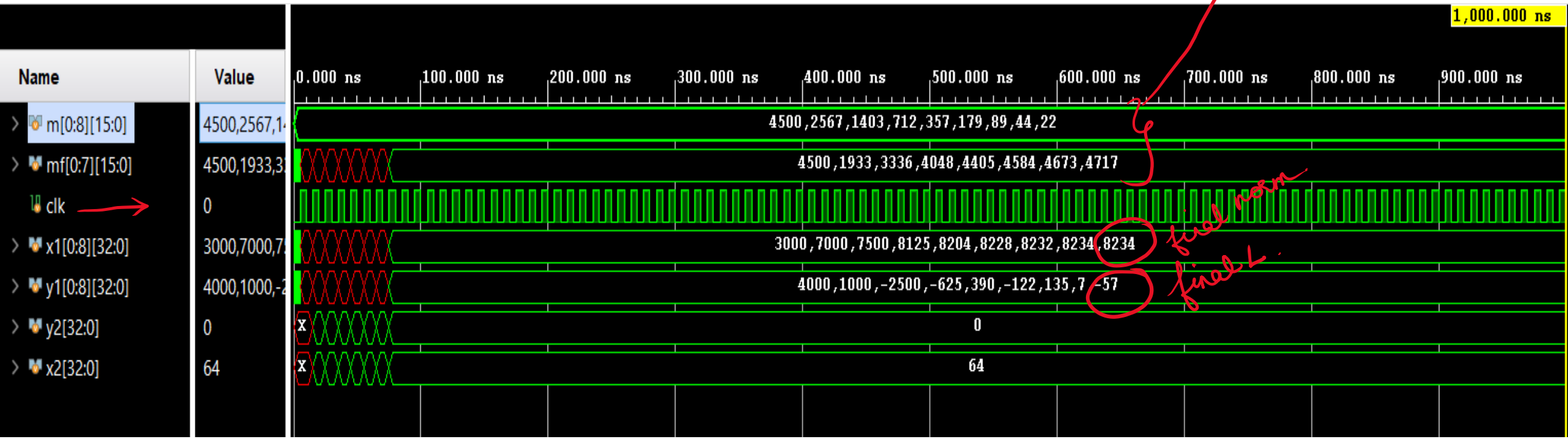
final values of y & x respectively in decimal

final y value

final x value

in hex

OUTPUT SIMULATIONS IN DECIMAL



0 iterations

final norm

final L

ITERATIONS

Name	Value	Data T...
[0][15:0]	4500	Array
[1][15:0]	2567	Array
[2][15:0]	1403	Array
[3][15:0]	712	Array
[4][15:0]	357	Array
[5][15:0]	179	Array
[6][15:0]	89	Array
[7][15:0]	44	Array
[8][15:0]	22	Array
mf[0:7][15:0]	4500,1933	Array
[0][15:0]	4500	Array
[1][15:0]	1933	Array
[2][15:0]	3336	Array
[3][15:0]	4048	Array
[4][15:0]	4405	Array
[5][15:0]	4584	Array
[6][15:0]	4673	Array
[7][15:0]	4717	Array

x1[0:8][32:0]	3000,7000	Array
[0][32:0]	3000	Array
[1][32:0]	7000	Array
[2][32:0]	7500	Array
[3][32:0]	8125	Array
[4][32:0]	8204	Array
[5][32:0]	8228	Array
[6][32:0]	8232	Array
[7][32:0]	8234	Array
[8][32:0]	8234	Array
y1[0:8][32:0]	4000,1000	Array
[0][32:0]	4000	Array
[1][32:0]	1000	Array
[2][32:0]	-2500	Array
[3][32:0]	-625	Array
[4][32:0]	390	Array
[5][32:0]	-122	Array
[6][32:0]	135	Array
[7][32:0]	7	Array
[8][32:0]	-57	Array

x value → norm → when y = 0

final L.

VALIDATIONS

(a) $\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$

$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = K_i R_{ot} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$

We begin with Anticlockwise & clockwise iterations respectively based on y being +ve or -ve until y becomes zero.

Anticlockwise	Clockwise
$x_{i+1} = x_i - y_i \delta_i 2^{-i}$	$x_{i+1} = x_i + y_i \delta_i 2^{-i}$
$y_{i+1} = y_i + x_i \delta_i 2^{-i}$	$y_{i+1} = y_i - x_i \delta_i 2^{-i}$

Respective Iterations

$i = 1$

initial values. $x_0 = 3$, $y_0 = 4$

$i = 1$

$\cancel{x_1 = 3 - 4(2^0) = 3 - 4 = -1}$

$\cancel{y_1 = 4 + 3(2^0) = 7}$

$x_1 = 3 + 4 = 7$

$y_1 = +1$

$y > 0$ clockwise

$(x_1, y_1) = (7, 1)$

$i = 2$

$x_2 = 7 + 1(1/2)$

$y_2 =$

$i = 3$

$x_3 = 7.5 - 2^{-2}(-2.5) = 8.125$

$y_3 = 2^{-2} \times 7.5 + (-2.5) = -0.625$

$y < 0$

anticlockwise

$i=4$ $x_4 = 8.125 - 2^{-3}(-0.625) = 8.203$

$y_4 = 2^{-3}(8.125) + (-0.625) = 0.390$

$y > 0$
clockwise

$i=5$ $x_5 = 8.203 + 2^{-4}(0.390) = 8.227$

$y_5 = -2^{-4}(8.203) + (0.390) = -0.122$

$y < 0$
anticlockwise

$i=6.$ $x_6 = 8.227 - 2^{-5}(0.22) = 8.230$

$y_6 = 2^{-5}(8.227) + (-0.122) = 0.135$

$y > 0$
clockwise

$i=7$ $x_7 = 8.230 + 2^{-6}(0.135) = 8.232$

$y_7 = -2^{-6}(8.230) + 0.135 = 0.006$

$y > 0$
clockwise

$i=8$ $x_8 = 8.232 + 2^{-7}(0.006) = 8.232$

$y_8 = -2^{-7}(8.232) + 0.006 = -0.058.$

Thus, Norm of $\vec{A}$ when $\vec{A} = [3, 4]$ is 8.232 i.e. the value of x at $y = 0$.

x

& Σ of all $\angle$'s uptill now gives $\theta = -57°$

θ

The code can be found here:
https://github.com/Amreen-Kaur/FPGA-LAB_IS21MTECH14002

THANKYOU