```
// Code your design here
`timescale 1ns/1ns

module rotational_mode
(
  input clk,
  input signed [15:0]theta,//angle
  input signed[15:0]x_in,
  input signed[15:0]y_in,
  output wire signed[19:0] x_final,y_final
);
  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


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

always @(posedge clk)
begin
  angle[0]<= (theta*256)>0 ? (theta*256)-micro_rotation[0] :
 (theta*256)+micro_rotation[0];
  s[0]<= theta*256>0 ? 1: -1;
  x[0]<=out0;
  y[0]<=out0_1;
  angle[1]<= (angle[0])>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]<= (angle[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]<=(angle[2])>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]<=(angle[3])>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]<=(angle[4])>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]<=(angle[5])>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]<=(angle[6])>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
//doing value c_inalc_inulation

assign x_final=0.607*out7;
assign y_final=0.607*out7_1;

endmodule
```

ADDER MODULE

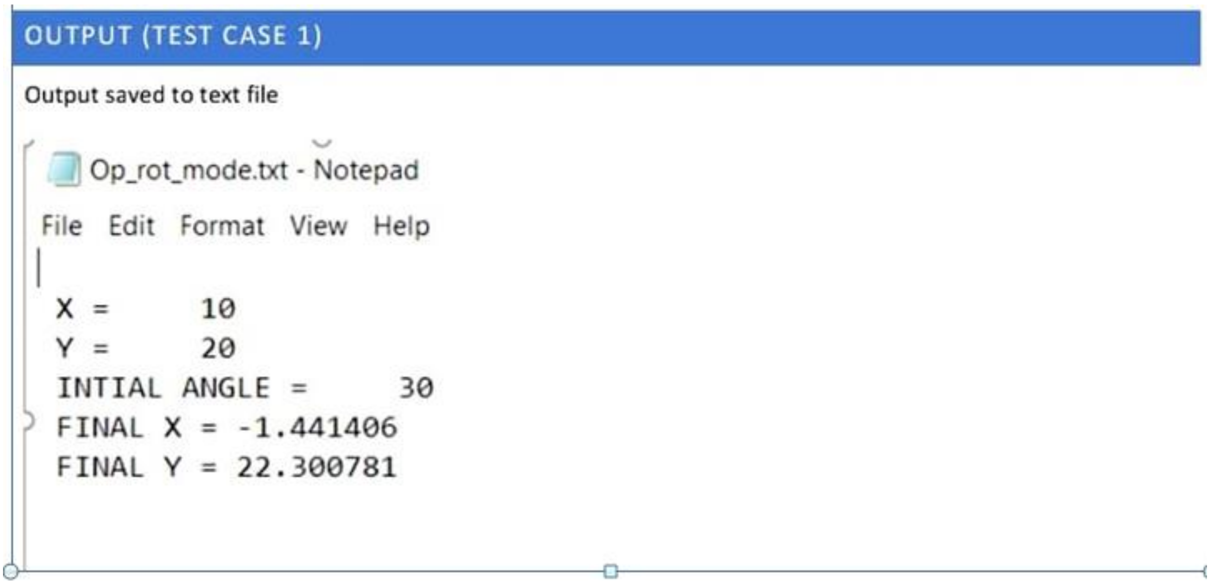
```
             module Adder(
               input signed[15:0] y1,
               input signed[15:0] y2,
               output wire signed[19:0] y
             );
               assign y=y1+y2;
endmodule
```

TEST BENCH :

```
`timescale 1ns/1ns

module test_bench;

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

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

  //instantiate the unit under test
rotational_mode uut(
    .clk(clk),
 .theta(theta),
 .x_in(x_in),
.y_in(y_in),
.x_final(x_final),
.y_final(y_final)
  );
integer  file_id;
localparam scale = 2**(-8.0);
real test_me;
```

```
 initial begin
clk=1;
  #400
  $display("\t \t VALUE  OF X= %d",x_in);
  $display("\t\t VALUE OF  Y = %d", y_in);
   $display("\t \t VALUE OF INITIAL  ANGLE=%d", (theta));
   $display("\t x_FINAL VALUE  OF X  =%d", (x_final));
   $display("\t \t FINAL VALUE  OF Y= %d", (y_final));
File_id=$fopen(“E:\\ IIT  Hyd\\semester2\\DIC\\rotation_mode_cordic\\op_rotation.text”);
begin
$fwrite(file_id,”\n  X=%d  \n Y=%d\n INITIAL ANGLE = %d \n  FINAL X = %f \n FINAL Y= %f”
(x_in),
(y_in),
(theta),
($itor(x_final*Scale)),
($itor(y_final*Scale)),
);


end
$fclose(file_id);
end




always #25 clk=~clk;


  initial begin
//TEST CASE 1
    theta = 15'd30;
x_in = 15'd10;
y_in = 15'd20;
//TEST CASE 2
```

```
theta =15’d40;

x__in=15’d10;

y_in = 15’d20;

//TEST case 3

theta = 15’d80;

x_in = 15’d30;

y_in = 15’d20;


  end


endmodule
```



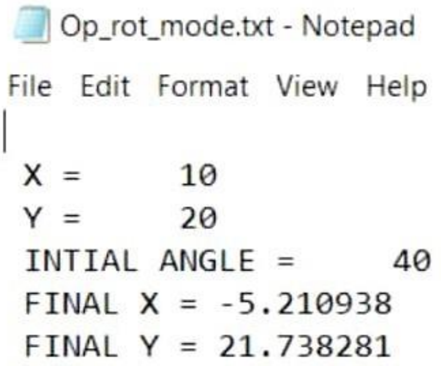
OUTPUT (TEST CASE 1)

Output saved to text file

```
X =      10
Y =      20
INTIAL ANGLE =      30
FINAL X = -1.441406
FINAL Y = 22.300781
```

OUTPUT (TEST CASE 2)

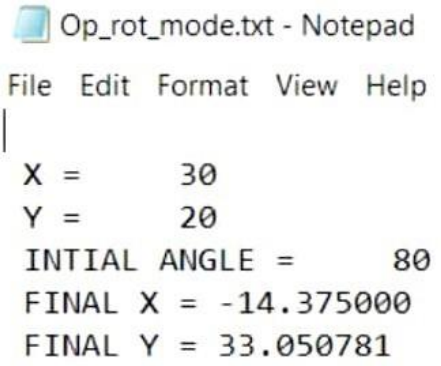
Output saved to text file



```
X =      10
Y =      20
INTIAL ANGLE =      40
FINAL X = -5.210938
FINAL Y = 21.738281
```

OUTPUT (TEST CASE3)

Output saved to text file



```
X =      30
Y =      20
INTIAL ANGLE =      80
FINAL X = -14.375000
FINAL Y = 33.050781
```