Project Report: Ultrasonic Radar

Muyi Chen

Kaiwen Ying

(Station 44)

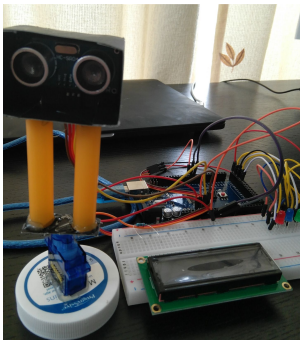
Project Introduction

The goal of the project is to design an ultrasonic radar which detects objects that are within the range of 40 centimeters from a spinning ultrasonic sensor. The data gathered by the sensor will be sent to be processed in an arduino board, and then to be displayed on FPGA with DEC_SOC. The project requires high level cooperation between the use of verilog and arduino. The arduino board is used to control the movement of the sonic sensor and to gather data, while the board DEC_SOC is used to control the FPGA, which displays the position data on the monitor.

In the brainstorming stage of the project, we looked up the internet and saw a variety of reference designs that could be implemented with hardware, and what really caught our eye was an FPGA controlled robot. We thought that combining a mechanical component with the FPGA would be more creative in a sense since most of the past projects have a software-like functionality. Furthermore, we think that a design like the ultrasonic radar has many real life applications and therefore would be worth doing.

Required hardware

1. DEC_SOC
2. Arduino Mega 2560
3. HC-SR04 ultrasonic sensor
4. SG90 servo
5. FPGA monitor
6. Wires
7. 5V power supply



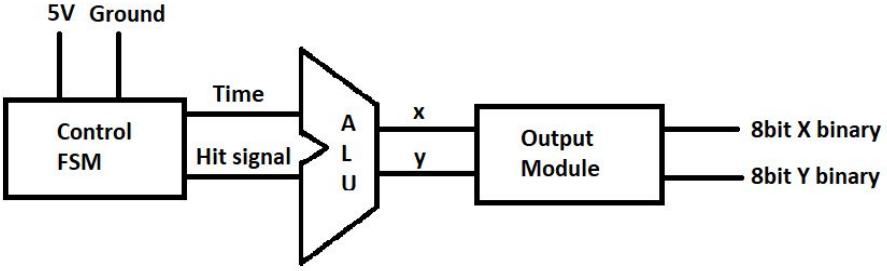
Detailed design

Arduino Part:

Block1 : Sonar Control FSM

Block2 : ALU

Block3 : Data Output



Block 1：Sonar Control FSM

This block controls the direction of the sonar sensor by driving the attached servo. The servo first spins for 180 degrees counter clockwise, and then clockwise until it reaches the starting position. Every step performed by the servo turns itself by 1 degree. When spinning counter clockwise, a for loop[1] is used to control the servo. Once 180 degree is reached, the next for loop[2] for clockwise movement is executed. In every loop, the data gathered by the sensor will be processed into a cartesian position (x,y), and then be computed into 5v signals into the DEC_SOC board.

Block2： ALU

The ALU plays the key part in the project. It first computes the distance data of the object[3]. The data gathered by the echo pin[4] of the sonar sensor is in the form of seconds. It displays the time used for the ultrasonic to hit the object and then bounce back to the sensor. Therefore the distance can be calculated by the formula Distance = time/2 * 340m/s [5], where time is the data from the sensor, and 340m/s is the general speed of the sound. The speed of sound will vary with temperature and humidity, but within a close range of 40 cm[6], the difference in speed of sound will not be a significant error.

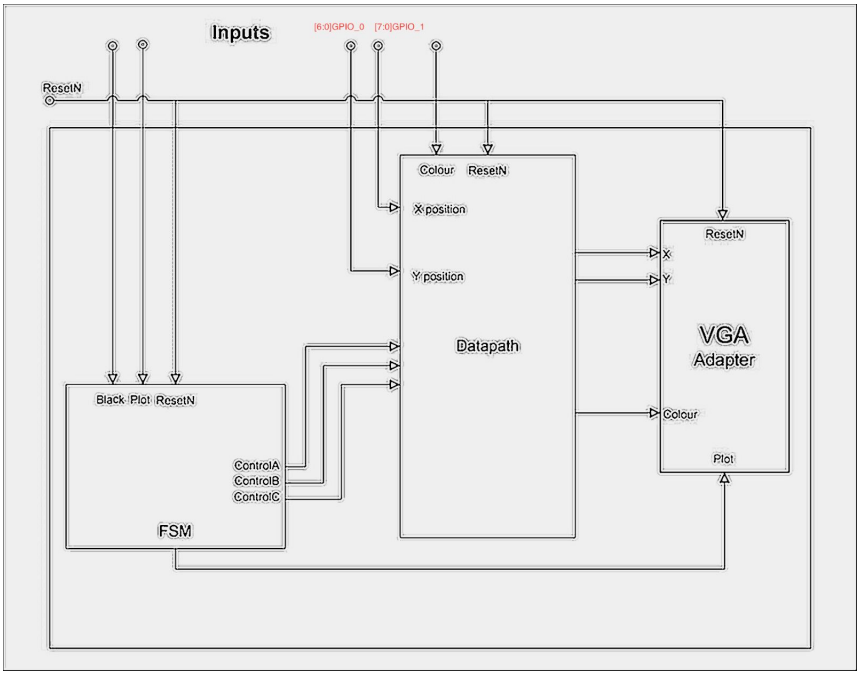
Then the ALU computes the actual x,y coordinates to be displayed on the monitor. The FPGA is takes in a 160x120 resolution signal, therefore the center position will be point (80,60), where x0 = 80, y0 = 60. With this original point, we can calculate the position data using the conversion from polar to cartesian coordination. The for loop[1] of the Control system will obtain an angle theta(i) which records the angle of the servo, therefore x = x0 + distance*cos(theta), and y = y0 + distance*sin(theta) [7]. The the x and y data will be further processed.

Block3 : Data Output

The final data output will be transferred by wire into the DEC_SOC board, therefore the coordinates must be converted into binary numbers before output. The x, y position from the ALU is an integer. When converting from decimal to binary, we obtain the method of modding by 2. The result of every modding operation(should be either 0 or 1) is then inserted as the first character of a string [8]. This string will be binary output of the position data. The maximum output distance is 40cm, therefore the maximum x value will be 120, which in binary is 1111000. To make it easier for further operations, all position data are modified into a 8 bit binary number.[9]

The final step of this block will be setting the output signals. The signals must be an 1 bit binary number, not a string. To solve this, every character in the string is iterated out first. Then an if statement[10] will be operated to verify whether this character is 0 or 1. The presetted output pin will be set high(5V) if the corresponding character is 1. [11] After all if the 16 declared output pins [12] are set up, the program delays [13] the output for 100ms for the DEC_SOC board to catch the signal, and then set all output pins to low(0V). [14] When this block finishes execution, the programs is set back to block 1. [15]

FPGA Component



In the verilog component, the signals generated by the Arduino program is then received and processed using the FPGA. The FPGA takes in an X coordinate and a Y coordinate from the pins [6:0]GPIO_1 and [6:0]GPIO_0 respectively and then draws a single pixel of color 3'b100 (red) on the VGA display. A low frequency clock at 10 Hz is included along with a D flip-flop in order to prevent the program from getting bits of incomplete data from the Arduino. The data first goes through the D flip-flop and is then connected to the VGA instance as two wires, which represent X and Y respectively. A preset 3'b100 wire is also connected to provide the colour.

Report on Success:

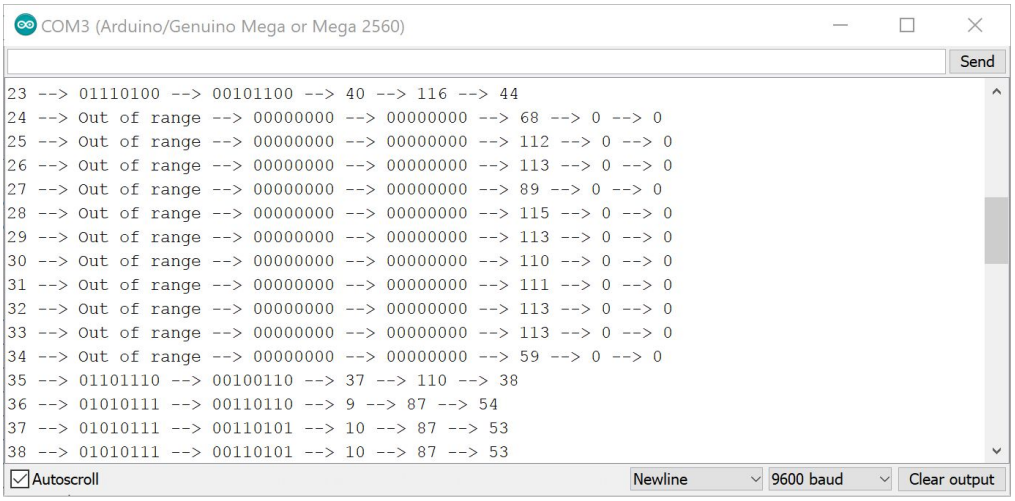
Arduino:

The arduino board with the sonar sensor works perfectly.

1.From arduino inspector:

1.The inspector shows the output of the data in the form of:
Angle → binary x → binary y → distance(cm) → x → y

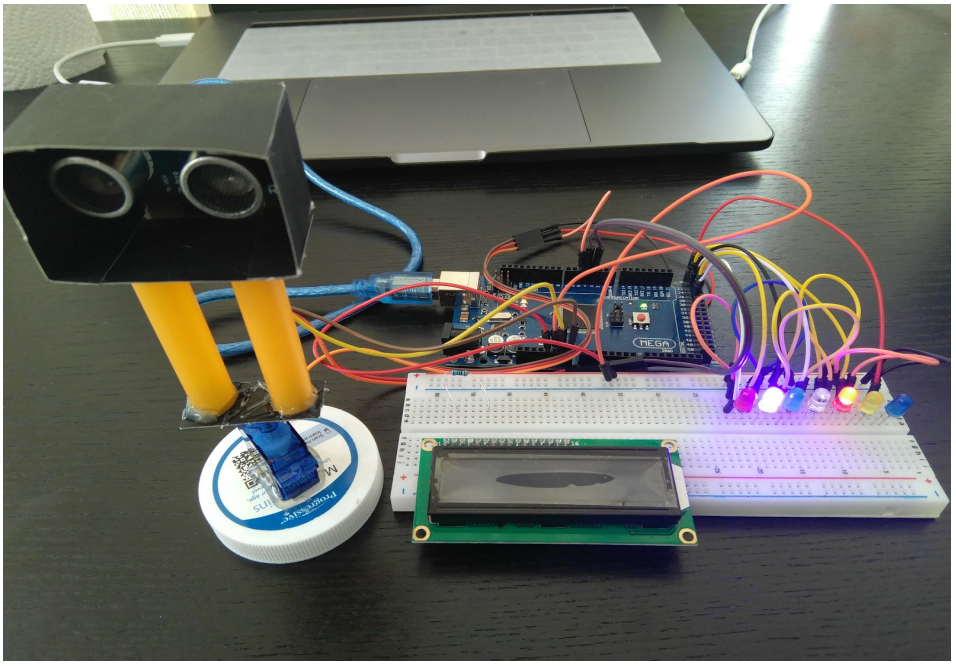
2.If the distance is bigger than 40 cm, the output will be "Out of Range" with zeros for x and y.



2.From the LED outputs:

1.7 LEDs are used to verify the binary output of the arduino board.
The LEDs should be lighted if signal “1” is the output.

2.The LED output of the board matches the binary number being calculated.



FPGA:

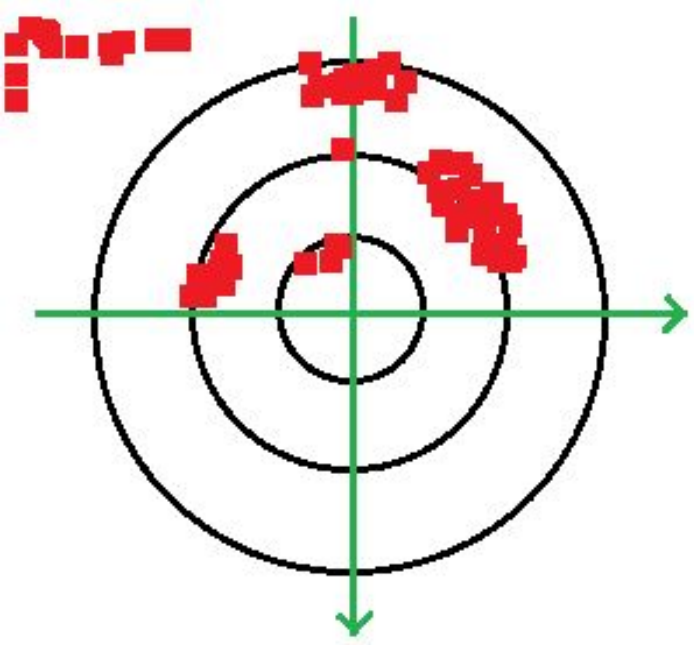
1.The “draw” program works, but some noise is present.

2.The VGA display does not get reset until the user presses KEY[0].

3.The screen displays the approximate shape of the obstacle.

4.Every (x,y) has been successfully drawn as a 4x4 red rectangle.

5.Some noise on the top left corner has been detected.



Improvements:

1. A new coordination on FPGA will be used. Due to the length of wire, the sensor can only spin for 180 degrees. Therefore a 180 degree coordinate can be used to improve visual impact.
2. Sound indication can be added when the ultrasonic wave hits an object.
3. Accuracy can be improved. For example, temperature and humidity sensors can be added to the system in order to accurately calculate the speed of sound.
4. A more stable low frequency clock can be included in order to decrease the amount of noise.
5. Can include an auto-clear function to keep the display updated on the location of obstacles.
6. The coordinate output frequency can be increased to allow the drawing process to be smoother.
7. A higher resolution display can be used.

Appendix

Arduino Code

```
//declaration
#include <Servo.h>
#include <math.h>
const int trigPin = 2;
const int echoPin = 3;
long duration;
int distance;
int xpos;
int ypos;
String resultx = "";
String resulty = "";
// loadx
int wir15 = 22;
int wir14 = 24;
int wir13 = 26;
int wir12 = 28;
int wir11 = 30;
int wir10 = 32;
int wir9 = 34;
int wir8 = 36;
// loady
int wir7 = 31;
int wir6 = 33;
int wir5 = 35;
int wir4 = 37;
int wir3 = 39;
int wir2 = 41;
int wir1 = 43;
int wir0 = 45;

//call servo
Servo myServo;
void setup(){
  //set up sonic sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  //attach servo to pin
  myServo.attach(4);
  //call serial to check data
  Serial.begin(9600);
 //set up outputs of the position data for verilog
```

[12]

```
  pinMode(wir15, OUTPUT);
  pinMode(wir14, OUTPUT);
  pinMode(wir13, OUTPUT);
  pinMode(wir12, OUTPUT);
  pinMode(wir11, OUTPUT);
  pinMode(wir10, OUTPUT);
  pinMode(wir9, OUTPUT);
  pinMode(wir8, OUTPUT);
  pinMode(wir7, OUTPUT);
  pinMode(wir6, OUTPUT);
  pinMode(wir5, OUTPUT);
  pinMode(wir4, OUTPUT);
  pinMode(wir3, OUTPUT);
  pinMode(wir2, OUTPUT);
  pinMode(wir1, OUTPUT);
  pinMode(wir0, OUTPUT);
}
void loop(){                                                    [15]
  xpos = 0;
  ypos = 0;

//controls the servo to spin from 0 to 180 degrees
for(int i=0;i<=180;i++){                                        [1]
  distance = 0;

  myServo.write(i);
  delay(10);
  distance = calculateDistance();                               [3]
  xpos = 80 + distance*cos(i*3.1415926/180);                    [7]
  ypos = 60 - distance*sin(i*3.1415926/180);
  int disx = xpos;
  int disy = ypos;
  int dis = distance;
  Serial.print(i);
  Serial.print(" --> ");
  if(distance > 40){                                            [6]
    Serial.print( "Out of range --> ");
    resultx = "00000000";
    disx = 0;
    disy = 0;
    resulty = "00000000";
  }
  else{
    // xpos in binary
    resultx = "";                                               [8]
    while(xpos > 0){
```

```
    if((xpos % 2) == 0){
      resultx = "0" + resultx;
    }else{
      resultx = "1" + resultx;
    }
    xpos /= 2;
  }
  String temp = "";
  if (resultx != "00000000"){
     while (resultx.length() < 8){
     temp = temp + "0";
     resultx = temp + resultx;
     temp = "";
    }
    }

  // ypos in binary
  resulty = "";
  while(ypos > 0){
    if((ypos % 2) == 0){
      resulty = "0" + resulty;
    }else{
      resulty = "1" + resulty;}
    ypos /= 2;
  }
 String tempy = "";

while (resulty.length() < 8){                                                    [9]
   tempy = tempy + "0";
   resulty = tempy + resulty;
   tempy = "";
}

}
Serial.print(resultx);
Serial.print(" --> ");
Serial.print(resulty);
Serial.print(" --> ");
Serial.print(distance);
Serial.print(" --> ");
Serial.print(disx);
Serial.print(" --> ");
Serial.print(disy);
Serial.print('\n'); /

////////////////////////////////////////  make binary outputs
```

```
//x with pins declared                                                    [10]
  if (resultx.charAt(7) == '1'){
  digitalWrite(wir15, HIGH);                                              [11]
}
   if (resultx.charAt(6) == '1'){
  digitalWrite(wir14, HIGH);
}
   if (resultx.charAt(5) == '1'){
  digitalWrite(wir13, HIGH);
}
   if (resultx.charAt(4) == '1'){
  digitalWrite(wir12, HIGH);
}
   if (resultx.charAt(3) == '1'){
  digitalWrite(wir11, HIGH);
}
   if (resultx.charAt(2) == '1'){
  digitalWrite(wir10, HIGH);
}
   if (resultx.charAt(1) == '1'){
  digitalWrite(wir9, HIGH);
}
   if (resultx.charAt(0) == '1'){
  digitalWrite(wir8, HIGH);
}


//y with pins declared
   if (resulty.charAt(7) == '1'){
  digitalWrite(wir7, HIGH);
}
   if (resulty.charAt(6) == '1'){
  digitalWrite(wir6, HIGH);
}

   if (resulty.charAt(5) == '1'){
  digitalWrite(wir5, HIGH);
}
   if (resulty.charAt(4) == '1'){
  digitalWrite(wir4, HIGH);
}
   if (resulty.charAt(3) == '1'){
  digitalWrite(wir3, HIGH);
}
   if (resulty.charAt(2) == '1'){
```

```
  digitalWrite(wir2, HIGH);
 }
   if (resulty.charAt(1) == '1'){
  digitalWrite(wir1, HIGH);
 }
   if (resulty.charAt(0) == '1'){
  digitalWrite(wir0, HIGH);
 }

 delay(100);                                                    [13]

//clear all signals to 0V                                      [14]
 digitalWrite(wir15, LOW);
 digitalWrite(wir14, LOW);
 digitalWrite(wir13, LOW);
 digitalWrite(wir12, LOW);
 digitalWrite(wir11, LOW);
 digitalWrite(wir10, LOW);
 digitalWrite(wir9, LOW);
 digitalWrite(wir8, LOW);

 digitalWrite(wir7, LOW);
 digitalWrite(wir6, LOW);
 digitalWrite(wir5, LOW);
 digitalWrite(wir4, LOW);
 digitalWrite(wir3, LOW);
 digitalWrite(wir2, LOW);
 digitalWrite(wir1, LOW);
 digitalWrite(wir0, LOW);

 //reset disx,disy
  disx = 0;
  disy = 0;
 }

 // from 180 to 0 degrees
 for(int i=180;i>0;i--){                                        [2]
 myServo.write(i);
 delay(16);
 distance = calculateDistance();
 xpos = 80 + distance*cos(i*3.14/180);
  ypos = 60 - distance*sin(i*3.14/180);
  int disx = xpos;
  int disy = ypos;
int dis = distance;
 Serial.print(i); // Sends the current degree into the Serial Port
```

```
  Serial.print(" --> "); // Sends addition character right next to the previous value needed later
in the Processing IDE for indexing
  if(distance > 40){
    Serial.print( "Out of range --> ");
    resultx = "00000000";
    disx = 0;
    disy = 0;
    resulty = "00000000";

  }
  else{
    // xpos
    resultx = "";
    while(xpos > 0){
      if((xpos % 2) == 0){

        resultx = "0" + resultx;

      }else{
        resultx = "1" + resultx;
      }
      xpos /= 2;
    }
    String temp = "";
    if (resultx != "00000000"){
       while (resultx.length() < 8){
       temp = temp + "0";
       resultx = temp + resultx;
       temp = "";
      }
      }
  // ypos
    resulty = "";
    while(ypos > 0){
      if((ypos % 2) == 0){
        resulty = "0" + resulty;
      }else{
        resulty = "1" + resulty;}
      ypos /= 2;
    }
   String tempy = "";

  while (resulty.length() < 8){
     tempy = tempy + "0";
     resulty = tempy + resulty;
     tempy = "";
```

```
}

}
Serial.print(resultx);
Serial.print(" --> ");
Serial.print(resulty);
Serial.print(" --> ");
Serial.print(distance);
Serial.print(" --> ");
Serial.print(disx);
Serial.print(" --> ");
Serial.print(disy);
Serial.print('\n');

////////////////////////////////////////// x

if (resultx.charAt(7) == '1'){
  digitalWrite(wir15, HIGH);
}
   if (resultx.charAt(6) == '1'){
  digitalWrite(wir14, HIGH);
}
   if (resultx.charAt(5) == '1'){
  digitalWrite(wir13, HIGH);
}
   if (resultx.charAt(4) == '1'){
  digitalWrite(wir12, HIGH);
}
   if (resultx.charAt(3) == '1'){
  digitalWrite(wir11, HIGH);
}
   if (resultx.charAt(2) == '1'){
  digitalWrite(wir10, HIGH);
}
   if (resultx.charAt(1) == '1'){
  digitalWrite(wir9, HIGH);
}
   if (resultx.charAt(0) == '1'){
  digitalWrite(wir8, HIGH);
}

////////////////////////////////////  y
   if (resulty.charAt(7) == '1'){
  digitalWrite(wir7, HIGH);
}
   if (resulty.charAt(6) == '1'){
```

```
  digitalWrite(wir6, HIGH);
}

   if (resulty.charAt(5) == '1'){
  digitalWrite(wir5, HIGH);
}
   if (resulty.charAt(4) == '1'){
  digitalWrite(wir4, HIGH);
}
   if (resulty.charAt(3) == '1'){
  digitalWrite(wir3, HIGH);
}
   if (resulty.charAt(2) == '1'){
  digitalWrite(wir2, HIGH);
}
   if (resulty.charAt(1) == '1'){
  digitalWrite(wir1, HIGH);
}
   if (resulty.charAt(0) == '1'){
  digitalWrite(wir0, HIGH);
}

delay(100);
//clear all
digitalWrite(wir15, LOW);
digitalWrite(wir14, LOW);
digitalWrite(wir13, LOW);
digitalWrite(wir12, LOW);
digitalWrite(wir11, LOW);
digitalWrite(wir10, LOW);
digitalWrite(wir9, LOW);
digitalWrite(wir8, LOW);

digitalWrite(wir7, LOW);
digitalWrite(wir6, LOW);
digitalWrite(wir5, LOW);
digitalWrite(wir4, LOW);
digitalWrite(wir3, LOW);
digitalWrite(wir2, LOW);
digitalWrite(wir1, LOW);
digitalWrite(wir0, LOW);

xpos = 0;
ypos = 0;
}
```

```
}
// calculate the distance of the object from the sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  //Make the trigPin high for 10 ms to send signal
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  //gets the time used
  duration = pulseIn(echoPin, HIGH);                                    [4]
  //calculate distance, note unit conversion
  distance= duration*0.034/2;                                           [5]
  return distance;
}
```

Verilog code

```
// Part 2 skeleton
`timescale 1ns/1ns

module Project
        (
                CLOCK_50,                               //      On Board 50 MHz
                // Your inputs and outputs here
                GPIO_0,KEY,
                // The ports below are for the VGA output.  Do not change.
                VGA_CLK,                                //      VGA Clock
                VGA_HS,                                         //      VGA
H_SYNC
                VGA_VS,                                         //      VGA
V_SYNC
                VGA_BLANK_N,                                    //      VGA
BLANK
                VGA_SYNC_N,                                     //      VGA
SYNC
                VGA_R,                                  //      VGA Red[9:0]
                VGA_G,                                          //      VGA
Green[9:0]
                VGA_B                                   //      VGA Blue[9:0]
```

```
	);

	input		CLOCK_50;			//	50 MHz
	// Declare your inputs and outputs here
	input [11:0]GPIO_0;//9-7clr,
	input [3:0]KEY;
	// Do not change the following outputs
	output		VGA_CLK;			//	VGA Clock
	output		VGA_HS;				//	VGA
H_SYNC
	output		VGA_VS;				//	VGA
V_SYNC
	output		VGA_BLANK_N;			//	VGA
BLANK
	output		VGA_SYNC_N;			//	VGA
SYNC
	output	[7:0]	VGA_R;			//	VGA Red[9:0]
	output	[7:0]	VGA_G;			//	VGA Green[9:0]
	output	[7:0]	VGA_B;			//	VGA Blue[9:0]

	wire resetn;
	assign resetn = KEY[0];

	// Create the colour, x, y and writeEn wires that are inputs to the controller.

	wire [2:0] colour;
	wire [7:0] x;
	wire [6:0] y;
	wire writeEn;
	//_____________counter
	reg CLOCK_Q;

	reg [2:0]counter;
	 always @(posedge CLOCK_50)
	begin
	counter <= counter + 1'b1;

	if(counter>=(3'b111))
		begin
		counter <= 3'b000;
		CLOCK_Q = 1;
		end
	else
		CLOCK_Q = 0;
	end//end of counter
```

```
//=======================================================================
==================
	//Need a 1Hz counter
	reg [26:0]hzcounter;
	reg [5:0]secondcount; //Stores the number of seconds passed since the last clear
	always @(posedge CLOCK_50)
	begin
		hzcounter <= hzcounter + 1'b1;

		if (hzcounter >= (8'd50000000))
		//When the 50MHz clock switches 5e7 times (takes 1 second)
		begin
			hzcounter <= 26'b00000000000000000000000000;
			secondcount <= hzcounter + 1'b1;
		end
	end//End of 1Hz counter

	reg onecycle;
	//Has value 1 if the sensor goes around 1 time
	always @(posedge CLOCK_50)
	begin
		if (secondcount >= (2'd30)) begin //The time for 1 revolution can be reset here
			onecycle <= 1;
		end

		else begin
			onecycle <= 0;
		end
	end

//=======================================================================
============================

	// Create an Instance of a VGA controller - there can be only one!
	// Define the number of colours as well as the initial background
	// image file (.MIF) for the controller.
	vga_adapter VGA(
			.resetn(resetn),
			.clock(CLOCK_50),
			.colour(3'b100),
			.x(x),
			.y(y),
			.plot(writeEn),
			/* Signals for the DAC to drive the monitor. */
```

```
                .VGA_R(VGA_R),
                .VGA_G(VGA_G),
                .VGA_B(VGA_B),
                .VGA_HS(VGA_HS),
                .VGA_VS(VGA_VS),
                .VGA_BLANK(VGA_BLANK_N),

                .VGA_SYNC(VGA_SYNC_N),
                .VGA_CLK(VGA_CLK));
        defparam VGA.RESOLUTION = "160x120";
        defparam VGA.MONOCHROME = "FALSE";
        defparam VGA.BITS_PER_COLOUR_CHANNEL = 1;
        defparam VGA.BACKGROUND_IMAGE = "image.mif";

    // Put your code here. Your code should produce signals x,y,colour and writeEn
    // for the VGA controller, in addition to any other functionality your design may
require.



    wire Xenable,Yenable;//from fsm to datpath
    wire[3:0] counterwire;
    wire VGAblank;


    FSM machine (.black(~KEY[2]),
                .plot(~KEY[1]),
                .endX(~KEY[3]),
                .resetN(KEY[0]),
                .Xenable(Xenable),
                .Yenable(Yenable),
                .plotout(writeEn),
                .counter(counterwire[3:0]),
                .clockQ(CLOCK_Q),
                .VGAblank(VGAblank),
                .onecycle(onecycle),
                );


    datapath path(.colour(3'b100),
                        .resetN(KEY[0]),
                        .switchinX(GPIO_0[6:0]),
                        .switchinY(GPIO_0[11:7]),
                        .Xenable(Xenable),
                        .Yenable(Yenable),
```

```
                                    .counter(counterwire[3:0]),
                                    .Xout(x[7:0]),
                                    .Yout(y[6:0]),
                                    .Cout(colour[2:0]),
                                    .VGAblank(VGAblank),
                                    .clock_Q(CLOCK_Q)
                                    );



endmodule//=========== end of top module ==============



module FSM (black,plot,endX,resetN,
Xenable,Yenable,plotout,counter,clockQ,VGAblank,onecycle);
        input black,plot,resetN,endX,clockQ,onecycle;
        output reg Xenable;
    output reg Yenable;
        output reg [3:0]counter;
        output reg plotout;
        output reg VGAblank;
        localparam loadX = 2'b00, loadY = 2'b01, clear = 2'b10, draw = 2'b11;

        reg [1:0] curS,nxtS;//state


        reg [15:0] counterB;

        always @(posedge clockQ)
        begin
                case(curS)
                default:begin curS = loadX;
                                        nxtS = loadX;

                                        end

                loadX: begin
                                        VGAblank <= 0;
                                        plotout = 0;
                                        Xenable = 1;

                                        if (endX == 1)
```

```
                        nxtS = loadY;
                        else if (black == 1)
                        nxtS = clear;
                        else
                        nxtS = loadX;

                end

        loadY: begin
                Xenable = 0;
                Yenable = 1;
                if(plot == 1)
                        begin
                        nxtS = draw;
                        Yenable = 0;
                        end
                else if (black == 1)
                        nxtS = clear;
                else
                nxtS = loadY;
                end

        clear: begin
                VGAblank <= 1;
                plotout = 1;
                if(counterB[15:0] <= 16'd38400)
                        nxtS = clear;
                else if (onecycle == 1) begin      // Clears the screen if the
sensor finishes rotating 1 cycle
                        nxtS = clear;
                end
                else
                        begin
                        nxtS = loadX;
                        plotout = 0;
                        end
                end

        draw: begin
                if(counter[3:0] !=4'b1111)
                        begin
                        plotout = 1;
                        nxtS = draw;
                        end

                else
```

```
                                        begin
                                plotout = 0;
                                nxtS = loadX;
                                //counter[3:0] = 3'b000;
                                        end
                                end//draw
        endcase
        end



        always @ (negedge clockQ)
                begin

                if(resetN == 0)
                curS = loadX;
                else
                curS = nxtS;
                if(plotout == 1)
                begin
                counter <= counter+1;
                counterB <= counterB +1;
                end
                else
                begin
                counter = 4'b0000;
                counterB = 16'd0;
          end
        end


endmodule// end of FSM

module
datapath(colour,resetN,switchinX,switchinY,Xenable,Yenable,counter,VGAblank,Xout,Yout,
Cout,clock_Q);
        input [2:0]colour ;
        input resetN;
        input [6:0]switchinX;
        input [6:0]switchinY;
        input Xenable, Yenable;
        input [3:0]counter;
        input VGAblank,clock_Q;
        output [7:0]Xout;
        output [6:0]Yout;
        output [2:0]Cout;
```

```
reg [7:0]Xreg;
reg [6:0]Yreg;
reg [2:0]Creg;

always@(*)
	begin

	if (VGAblank ==1)
		begin
		Xreg[7:0]=blackX[7:0];
		Yreg[6:0]=blackY[6:0];
		end

	else if (Xenable == 1)
		begin
		Xreg[6:0] = switchinX[6:0];
		Xreg[7] = 0;
		end

	else if(Yenable == 1)
	begin
		Yreg[6:0] = switchinY[6:0];
		Creg[2:0] = colour[2:0];
	end

end

assign Xout[7:0] = (VGAblank == 0)? (Xreg[7:0]+counter[1:0]):Xreg[7:0];
assign Yout[6:0] = (VGAblank == 0)? (Yreg[6:0]+counter[3:2]):Yreg[6:0];
assign Cout[2:0] = (VGAblank == 0)? Creg[2:0]:3'b000;



wire [7:0]blackX;
wire [6:0]blackY;
//==================black============

clear_to_black u0 (.VGAblank(VGAblank),
                                                .reset(resetN),
                                                .clock(clock_Q),
                                                .xout(blackX[7:0]),
                                                .yout(blackY[6:0]),
                                                );


//end of black
```

```
endmodule


module clear_to_black (VGAblank, reset, clock, xout, yout, plot);
    input VGAblank, reset, clock;
output reg[7:0] xout;
output reg[6:0] yout;
output reg plot;

reg[15:0] counter;

always@(posedge clock)
begin
 /*   if (!reset)
 begin
 counter = 15'b0;
 end
 else*/ if (VGAblank == 1 & counter <= 16'd38400)
 begin
     xout = counter[7:0];
yout = counter[15:8];
counter = counter + 1;
//colour = 3'b000;
plot = 1;
 end
 else if (counter >16'd38400)
     counter = 16'd0;
end

endmodule
```