EE536P - FPGA Lab
Programming Assignment : Waveform Generation Based on CORDIC and FPGA

1 Introduction

In this assignment, the objective is sine wave generation and display. A CORDIC design (that does not pay attention to the scale factor) running on FPGA will be the basis for sine wave generation. The output will be directed to a PC monitor (via the VGA display port on the EDGE Artix board).

2 CORDIC for Sine Wave Generation

CORDIC in rotation mode will be used for waveform generation. Input angles in the range 0 to 90 are specified in increments of a degrees ($a * \Pi/180$ converted to hexadecimal in the Verilog program). It is advantageous to take total number of samples (of sine wave) in the range 0 to 360, as a power of 2. CORDIC will output the sine value which can be modified suitably for display on the PC monitor. One approach is to add/subtract a fixed value to the output of CORDIC computation. The approach chosen is "often" decided by the "medium" in which we want the results (in this case a PC monitor via the VGA port).

3 Sine Wave on a PC Monitor via the VGA Display Port

The Edge Artix FPGA board user guide (available via Google classroom course) has a section on *VGA Display Port*. Please go through this section so that the partial programs given below can be completed without difficulty.

4 Tasks to be Performed

Task 1: Complete the Verilog module below pertaining to unscaled CORDIC (unscaled is adequate for sine wave generation).

```
// Unscaled sine value generation
module CORDIC_sine(clk, start, angle, sint,  done);

input clk, start;
input [15:0] angle;  //2 bits for integer and 14 for fractional part
   //    adequate to consider maximum value of angle for sine as 90 degrees
  // angle = 90 * pi/180 =  01.10010010001000 = 16'h6488 (radians)


output reg [7:0] sint;  //2 bits integer, 6 bits fraction
output reg done;

reg [15:0]x0, y0; // 2 bits integer, 14 bits fraction
reg [15:0] sx, sy; //2.14
reg [15:0]z0;
reg [15:0]atan; // 2 bits integer, 14 bits fraction
reg d;
reg [4:0]i;

always@(posedge clk)
begin
if(start) begin  // Initialize
z0 =  angle;
x0 = 16'h4000;  // decimal value of 1.000
y0 = 16'd0;
 i = 5'd0;
 d = 1'b0;
 done = 1'b0;
```

```
end
else begin
if(i < 5'b10000) begin

// case module to implement  tan^-1(2^-i)
// Arithmetic right shift of  "x0" and "y0"; store in "sx" and "sy" respectively
// i.e sx = x0 * 2^-i
// i.e sy = y0 * 2^-i

// the atan values are stored  with 2-bit integer part and 14-bit fractional part
//tan^-1(2^-0) = 45 degrees = 45 * pi/180 = 00.11_0010_0100_0100 = 16'h3244
case(i)
5'd0: begin atan= 16'h..........;  // fill in for tan^-1(2^-0)  (in radians)
sx = x0;
sy = y0 ;    end
5'd1: begin atan= 16'h........;   // fill in for tan^-1(2^-1)  (in radians)
sx = {x0[15], x0[15:1] };
sy = {y0[15], y0[15:1] };    end
5'd2: begin atan= 16'h........; // fill in for tan^-1(2^-2)  (in radians)
sx = {.......... };
sy = {.......... };   end
5'd3: begin atan= 16'h....; // extend to other elementary angles
  sx = {.......... };
sy = {.......... };   end
5'd4: begin atan= 16'h....;
sx = {.......... };
sy = {.......... };   end
5'd5: begin atan= 16'h.......;
sx = {.......... };
sy = {.......... };   end
5'd6: begin atan= 16'h....;
sx = {.......... };
sy = {.......... };   end
5'd7: begin atan= 16'h.........;
sx = {.......... };
sy = {.......... };   end
5'd8: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd9: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd10: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd11: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd12: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd13: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd14: begin atan= 16'......;
sx = {.......... };
sy = {.......... };   end
5'd15: begin atan= 16'......;
sx = {.......... };
```

```
sy = {......... };   end
default: atan= 16'h0000;
endcase


// here d == 0  represents +1 and d== 1 represents -1
if(d == 1'b0)begin
x0 = ........;
y0 =.........;
z0 = ........;
end
else begin
x0 =......;
y0 =.......;
z0 = ......;
end

d = .......;// assume z0 is in 2's compl form ... use msb ..
done = 1'b0;
i = i + 1;
end
else begin
done = 1'b1;
sint = y0[15:8];
end

end
end
endmodule
```

Task 2: Write a Verilog testbench and simulate the CORDIC design for the full set of angles.

Task 3: Complete the code for the VGA display given below.

```
module sine_wave_from_CORDIC(mclk, start, hc, vc, vidon, red, grn, blu);

input mclk, start;

reg [7:0]sine_wave;

input [9:0]hc, vc;
input vidon;

output reg red, grn, blu;


reg start_cr;
reg [15:0]angle;
wire [7:0] sint;  //2.6
wire done;
reg  stop;

parameter N = 64;
parameter M = 128;

reg [8:0]count;

reg [9:0] ref_h_pix ;     // starting reference horizontal pixel value -- 72 < ref_h_pix < 784
reg [9:0] ref_v_pix ;     // starting reference vertical pixel value  --31 < ref_v_pix < 511

reg [N-1:0]lcd_pixel[0:M-1];

reg [N-1:0]temp;
reg [N-1:0]temp2;
reg [7:0]temp1;
reg start1;




 CORDIC_sine crd1( mclk, start_cr, angle, sint, done );

// 1 degree = 1 * pi/180 = 00.00_0001_0001_1110 = 011E

always@(negedge mclk)
begin
  if(start) begin
start_cr = 1;
angle = 16'h0000;
count = 8'd0;
 stop = 1'b0;
start1 = 1'b1;

end else begin
 if(done == 1'b1) begin
if(count <= 9'd15) begin
angle =  angle + 16'h06B4; // 6 degrees
count = count + 1'b1;
sine_wave = 8'h80 + sint;
stop =  1'b0;
end
else if(count > 9'd15 && count <= 9'd31)begin
angle = ........;
count = .........;
sine_wave = .............;
stop =  .........;
end
else if(count > 9'd31 && count <= 9'd47)begin
angle =  ..........;
count = ..........;
sine_wave = ..........;
           stop =  .......;
end
else if(count > 9'd47 && count < 9'd64)begin
angle =  ..........;
count = ..........;
sine_wave = ..........;
           stop =  .......;
end
else begin
angle = 16'h0000;
stop  = 1'b1;
start1 = 1'b0;
end
```

```
           // converting sine wave values into an ‘image’
           // lcd_pixel is a  two-dimensional binary matrix which
           // contains information on the pixel that needs to ‘glow’
           // a sample matrix is shown at the end of this handout
              temp = lcd_pixel[sine_wave[7:1]];
  temp[64-count] = 1’b1;
  lcd_pixel[sine_wave[7:1]] =  temp;

  start_cr = 1’b1;
  end
  else begin
if(stop)
start_cr = 1’b1;
else
start_cr = 1’b0;
  end
end
 end



// starting reference horizontal pixel value -- 72 < ref_h_pix < 784
// starting reference vertical pixel value  --31 < ref_v_pix < 511

// Use arguments  hc (horizontal count), vc (vertical count), vidon of  VGA_hs_vs module below for
// display of value

always@(posedge mclk)
begin
if(start1)
begin
ref_h_pix = 10’d 256; // reference values as a power of 2
ref_v_pix = 10’d 128;
end
else if(vidon)
begin
if((hc>ref_h_pix)&&(hc < .......... ) && (vc>ref_v_pix)&&(vc < ...........) )
begin
 temp2 = lcd_pixel[vc[6:0]];

    if(temp2[hc[5:0]] == 1’b1)
 grn = ..........;
else grn = ........;
end
else begin
 grn = ......;
     end


// Creating Horizontal and Vertical lines
if((hc == ref_h_pix ) || ( vc == ref_v_pix + 64))
blu = ....;
else
blu = .....;


red = .....;

end
```

```
   else
begin
grn = ....;
blu = .....;
red = ......;
   end


end
endmodule
```

Task 4: Copy the Verilog file called *VGA_hs_vs.v* that has a 'VGA enable' module from the Google classroom course. Invoke the module in the top level module given below. Then write a ".ucf" file and implement the design of CORDIC on the FPGA board and show the output on the VGA display.

```
module top_CORDIC_VGA(mclk, start, hs, vs,  red, grn, blu );

input mclk, start;
output hs, vs;
output  red, grn, blu;

wire [9:0] hc , vc ;  // These are the Horizontal and Vertical counters
wire vidon;   // Tells whether or not it is ok to display data


VGA_hs_vs  VHV1 (mclk, start, hs, vs, hc, vc, vidon);

   sine_wave_from_CORDIC SCV1(mclk, start, hc, vc, vidon, red, grn, blu);

endmodule
```

5 How can *lcd_pixel* be used to display a sine wave ?

Here is a sample assignment to *lcd_pixel*. Observe that the 1's together resemble a sine wave.

```
 lcd_pixel (16 x 8)

   0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
   0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
   0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0
   1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
   0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0
   0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0
   0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
```