# Experiment 3 - Function Generator

Ali Padyav, 810199388
Bita Nasiri, 810199504

***Abstract*— This document is a report about designing Function Generator and implementing on FPGA for the 3nd experiment of the DLD Lab (ECE 045) at University of Tehran, Department of Electrical and Computer Engineering.**

***Keywords*— Function Generator, PWM, DAC, FPGA Programming, Altera Cyclone II, Quartus, RTL Design**

## I. Waveform Generator

### A. Code of Waveform Generator

```
module sine_wave(input clk, rst, output [7:0] sin);
  reg signed [15:0] sin_t = 16'd0, cos_t = 16'd30000;
  reg signed [15:0] sin_r, cos_r;

  always @(posedge clk)
  begin
    if (rst)
    begin
      sin_t = 16'd0;
      cos_t = 16'd30000;
    end
    else
    begin
      sin_r = sin_t + (cos_t >>> 6);
      cos_r = cos_t - (sin_r >>> 6);
      sin_t = sin_r;
      cos_t = cos_r;
    end
  end

  assign sin = sin_r[15:8] + 8'd127;
endmodule
```

Fig. 1 Verilog description of sine wave

```
module half_wave(input [7:0] sine, output reg [7:0] out);
  always @(sine)
  begin
    if (sine < 8'd128)
      out = 8'd127;
    else
      out = sine;
  end
endmodule
```

Fig. 2 Verilog description of half wave

```
module full_wave(input [7:0] sine, output reg [7:0] out);
  always @(sine)
  begin
    if (sine < 8'd128)
      out = 8'd255 - sine;
    else
      out = sine;
  end
endmodule
```

Fig. 3 Verilog description of full wave

```
module sqaure_wave(input clk, rst, output reg [7:0] square);
  reg [7:0] count = 8'd0;

  always @(posedge clk)
  begin
    if (rst)
      count = 8'd0;
    else
      count = count + 1;

    if (count < 128)
      square = 8'd0;
    else
      square = 8'd255;
  end
endmodule
```

Fig. 4 Verilog description of square wave

```
module triangle_wave(input clk, rst, output reg [7:0] triangle);
  reg [7:0] count = 8'd0;

  always @(posedge clk)
  begin
    if (rst)
    begin
      triangle <= 8'b0;
      count <= 1'b0;
    end
    else
    begin
      count <= count + 1'b1;
      if (count < 8'd127)
        triangle <= triangle + 2'd2;
      else if(count == 8'd 127)
        triangle <= triangle + 2'd1;
      else if(count == 8'd255)
        triangle <= triangle - 2'd1;
      else
        triangle <= triangle - 2'd2;
    end
  end
endmodule
```

Fig. 5 Verilog description of triangle wave

```verilog
module reciprocal_wave(input clk, rst, output reg [7:0] out);
  reg [7:0] count = 8'd0;

  always @(posedge clk)
  begin
  out <= 8'b0;
    if (rst)
    begin
      out <= 8'b0;
      count <= 1'b0;
    end
    else
    begin
      count <= count + 1'b1;
      if (count >= 8'd252)
        out <= 8'd4;
      else
        out <= 8'd255 / (8'd63 - (count >> 2));
    end
  end
endmodule
```

Fig. 6 Verilog description of reciprocal wave

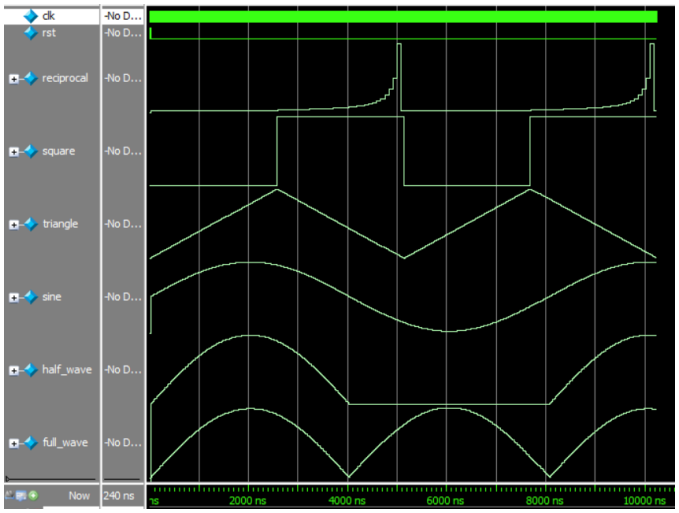


Fig. 7 Simulation of all waveforms

```verilog
module wave_generator(input clk, rst, output[7:0] reciprocal, square,
                      triangle, sine, half_wave, full_wave);

  reciprocal_wave rw1(clk, rst, reciprocal);
  sqaure_wave sqaure_w1(clk, rst, square);
  triangle_wave tw1(clk, rst, triangle);
  sine_wave sine_w1(clk, rst, sine);
  half_wave hw1(sine, half_wave);
  full_wave fw1(sine, full_wave);
endmodule
```

Fig. 6 Verilog description of whole waveform generator

## II. DIGITAL TO ANALOG CONVERSION USING PWM

This module convert digital waves to analog.

### A. Code of DAC

```verilog
module DAC(input clk, rst, input [7:0] wave, output reg pulse);
  reg[7:0] count = 8'd0;

  always @(posedge clk)
  begin
    if (rst)
      count = 8'd0;
    else
      count = count +  8'd1;
  end

  always @(count)
  begin
    pulse = (count < wave) ? 1'b1 : 1'b0;
  end
endmodule

```

Fig. 8 Verilog description of DAC model

### B. Block Diagram

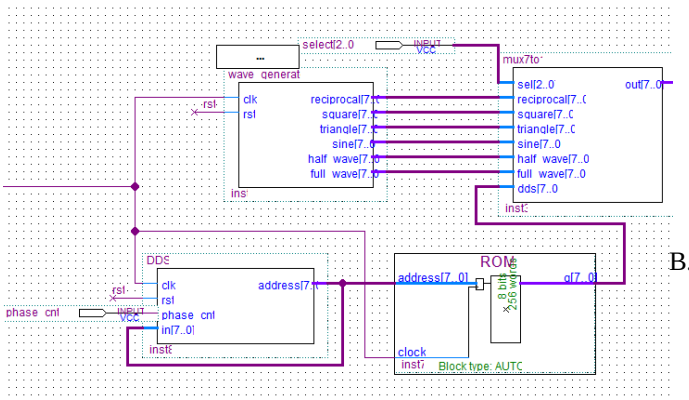


Fig. 7 Verilog description of whole waveform generator

We didn't set this as top module. Thus synthesis summery doesn't exist.

### C. Modelsin Results for all waveforms

### B. Simulation

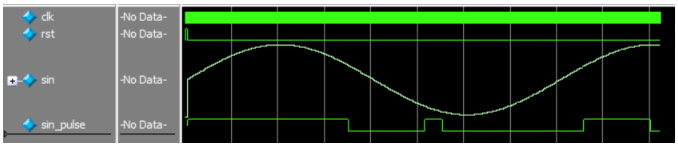


Fig. 9 Simulation of DAC module

### C. Program to FPGA

We add all modules to Quartus and synthesize them. Connected modules together and programmed this top module to FPGA.

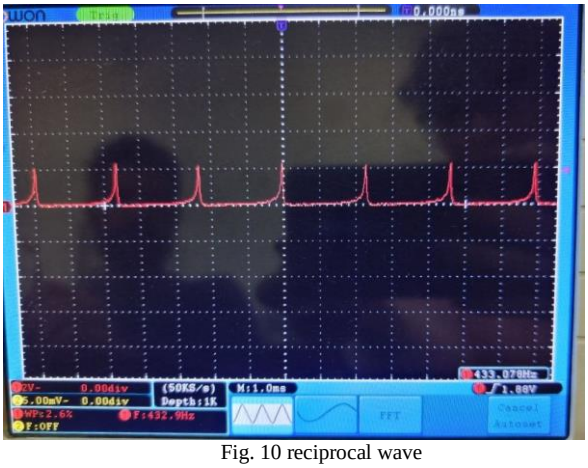

Fig. 10 reciprocal wave

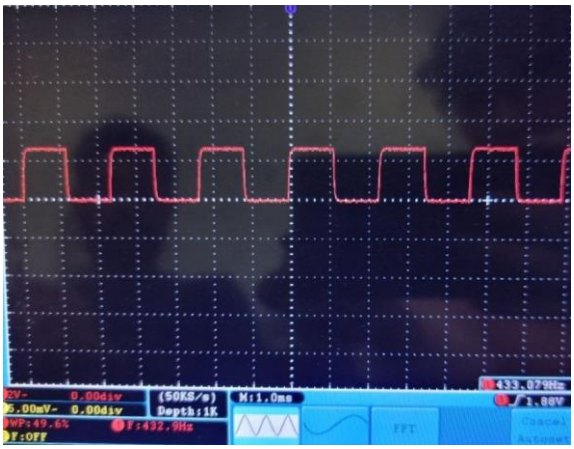

Fig. 11 square wave

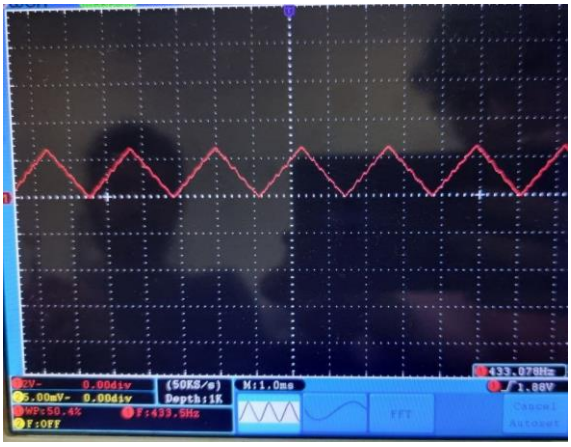

Fig. 12 triangle wave

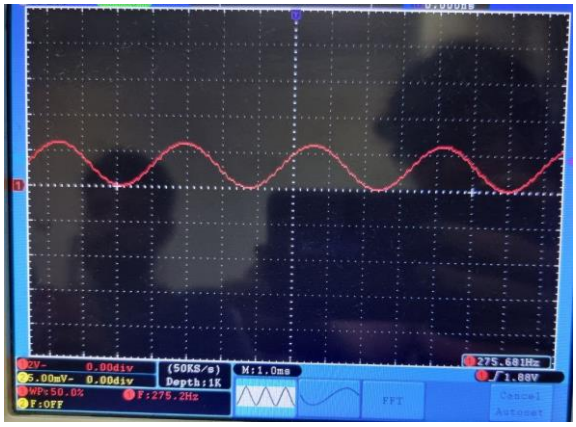

Fig. 13 sine wave

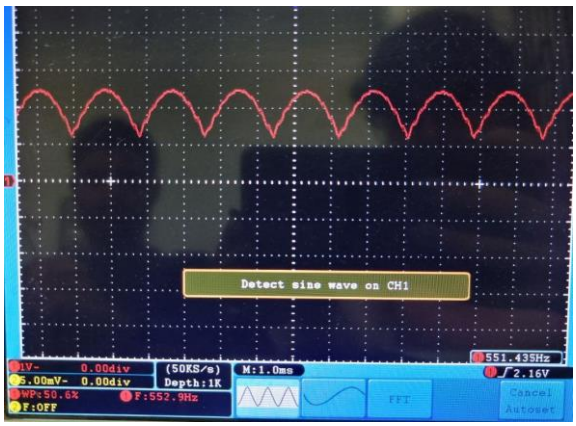

Fig. 14 full wave

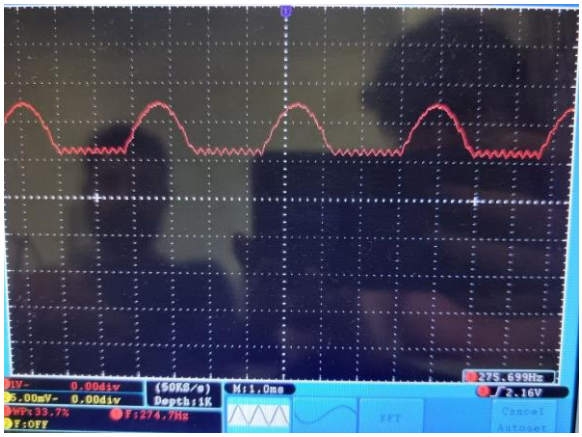

Fig. 15 half wave

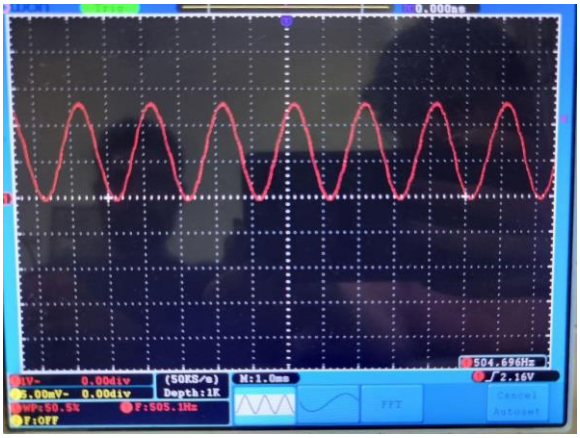


Fig. 16 sine wave with freq 110

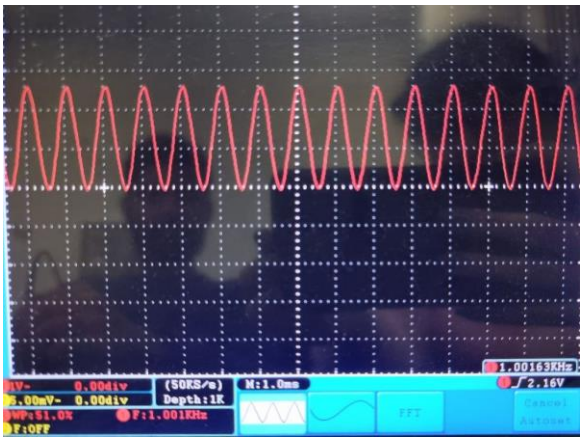


Fig. 17 sine wave with freq 111

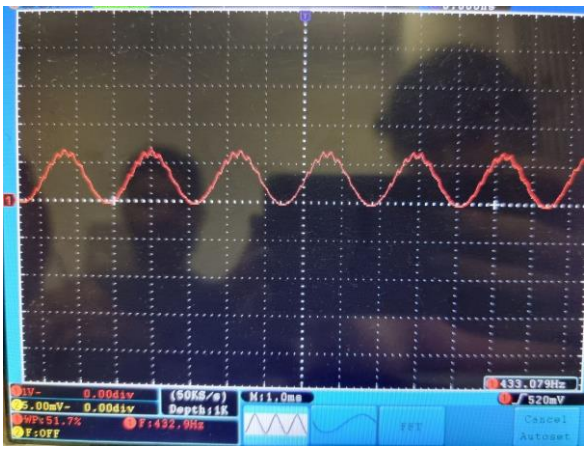


Fig. 18 sine wave with ampliude 10 that half domain

## III. Frequency Selector

### A. Test and Simulation

#### 1.Frequency Selector

In fig.19 freq = $10^9/5200$ = 192 KHz

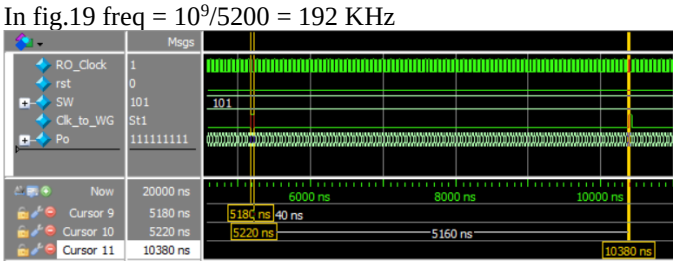


Fig. 19 sw = 101

In fig.20 freq = $10^9/2700$ = 370 KHz

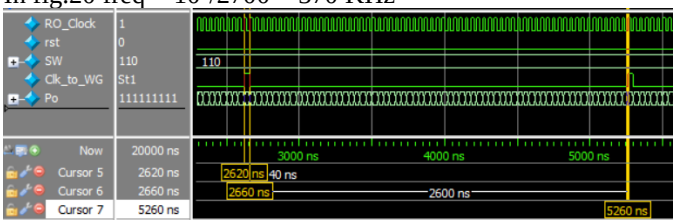


Fig. 20 sw = 110

In fig.21 freq = $10^9/80$ = 12 MHz

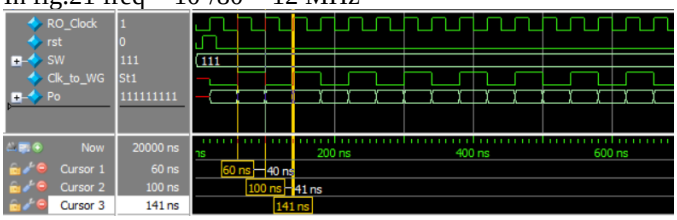


Fig. 21 sw = 111

#### 2. DDS

We set phase control to 1, 5 and 10

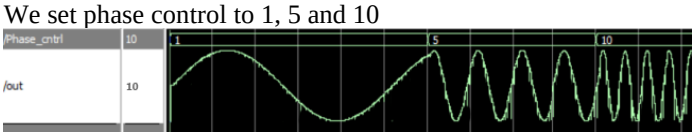


Fig. 22 waveform with different phase control of DDS

## IV. AMPLITUDE SELECTOR

### A. Codes

```
module amplitud_Selector(input [1:0] select, input [7:0] in, output [7:0] out);
  assign out = in >> select;
endmodule
```

Fig. 22 Verilog description of amlitude selector

```
1
2  module amp_TB();
3
4  reg clk = 1'b0, rst = 1'b1;
5  wire [7:0] wave, out;
6  reg [1:0] sel = 2'b00;
7
8  sine_wave sine(clk, rst, wave);
9
10 amplitud_Selector as(sel, wave, out);
11
12   initial
13   begin
14     repeat(7000)
15       #10 clk = ~clk;
16   end
17
18 initial begin
19     #10 rst = 1'b0;
20     #(2048 * 10) sel = 2'b01;
21     #(2048 * 10) sel = 2'b10;
22     #(2048 * 10) sel = 2'b11;
23 end
24
25 endmodule
```

Fig. 23 Testbench of amplitude selector

B. Test and Simulation

It is obvious that by increasing phase of amplitude domain of wave decreases.

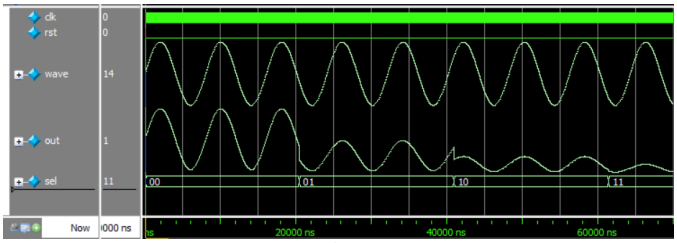


Fig. 24 Testbench of amplitude selector

## V. Total Design

We import Verilog description to quartus and create some modules like ROM by magic wizards. Then connect all these modules. Compiled it and assigned pins.

### A. RTL Design

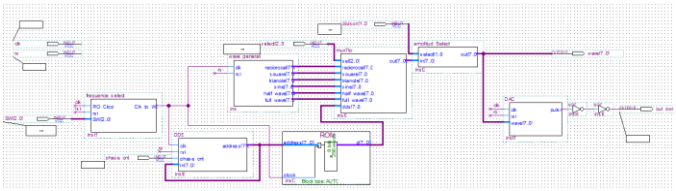

Fig. 25 Block diagram of whole function generator

### B. Pin Planner

| To | Direction | Location | I/O Bank | VREF Grou | Fitter Loca |
|---|---|---|---|---|---|
| clk | Input | PIN_L1 | 2 | B2_N1 | PIN_L1 |
| divison[1] | Input | PIN_U11 | 8 | B8_N0 | PIN_U11 |
| divison[0] | Input | PIN_U12 | 8 | B8_N0 | PIN_U12 |
| out_pwm | Output | PIN_A13 | 4 | B4_N1 | PIN_A13 |
| phase_cnt | Input | PIN_L21 | 5 | B5_N1 | PIN_L21 |
| rst | Input | PIN_L22 | 5 | B5_N1 | PIN_L22 |
| select[2] | Input | PIN_L2 | 2 | B2_N1 | PIN_L2 |
| select[1] | Input | PIN_M1 | 1 | B1_N0 | PIN_M1 |
| select[0] | Input | PIN_M2 | 1 | B1_N0 | PIN_M2 |
| SW[2] | Input | PIN_W12 | 7 | B7_N1 | PIN_W12 |
| SW[1] | Input | PIN_V12 | 7 | B7_N1 | PIN_V12 |
| SW[0] | Input | PIN_M22 | 6 | B6_N0 | PIN_M22 |
| wave[7] | Output | | | | PIN_A15 |
| wave[6] | Output | | | | PIN_H12 |
| wave[5] | Output | | | | PIN_G12 |
| wave[4] | Output | | | | PIN_F12 |
| wave[3] | Output | | | | PIN_B14 |
| wave[2] | Output | | | | PIN_B13 |
| wave[1] | Output | | | | PIN_C13 |
| wave[0] | Output | | | | PIN_A14 |