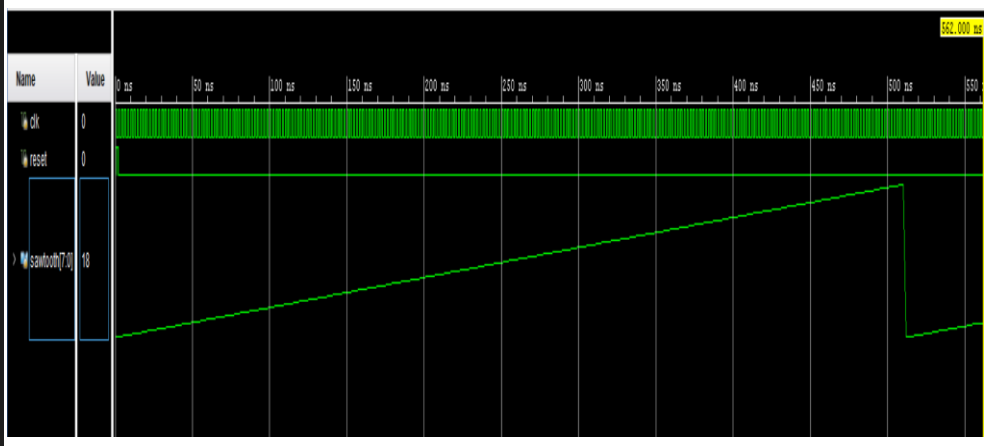
ASIC & FPGA HW 5

Parham Gilani – 400101859

سوال 1)

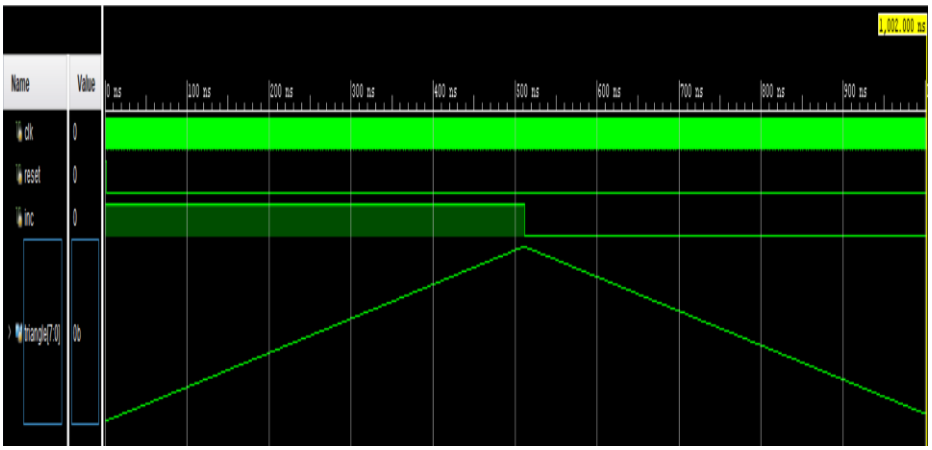
(a

```
module sawtooth_tb;

    reg clk, reset;
    reg [7:0] sawtooth;

    always @(clk) #1 clk <= ~clk;

    always @(posedge clk) begin
        if (sawtooth == 8'hFF || reset) sawtooth <= 8'h00;
        else sawtooth <= sawtooth + 1;
    end

    initial begin

        clk = 1;
        reset = 1;
        #2
        reset = 0;

        #560
        $stop;
    end

endmodule
```



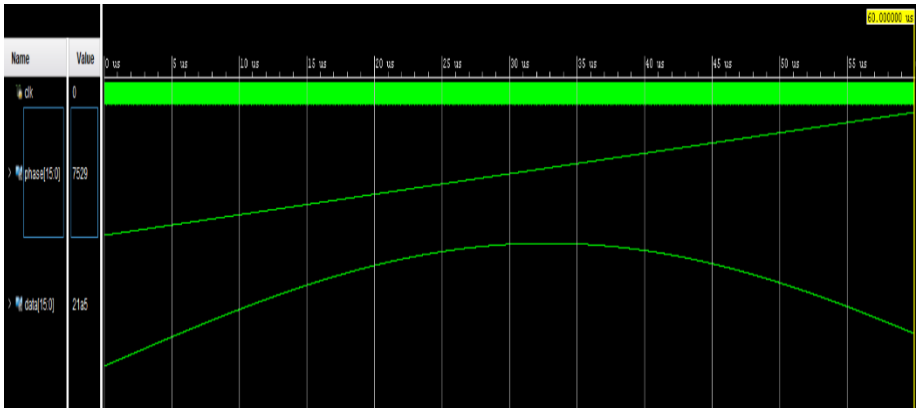
(b

```
module triangle_tb;

    reg clk, reset;
    reg [7:0] triangle;
    reg inc;

    always @(clk) #1 clk <= ~clk;

    always @(posedge clk) begin
        if (reset) triangle <= 8'h00;
        else begin
            if (inc && triangle != 8'hFF) triangle <= triangle + 1;
            if (triangle == 8'hFF) inc <= 0;
            if (~inc && triangle != 8'h00) triangle <= triangle - 1;
            if (triangle == 8'h00) inc <= 1;
        end
    end

    initial begin
        inc = 1;
        clk = 1;
        reset = 1;
        #2
        reset = 0;

        #1000
        $stop;
    end

endmodule
```



(c

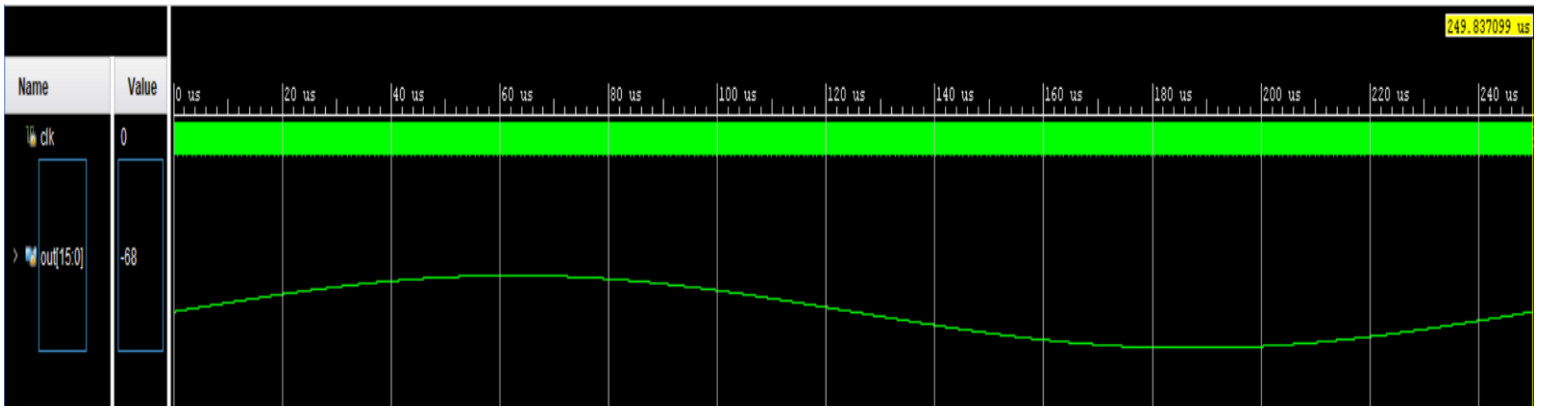
```
`timescale 1ns / 1ps

module dds_tb;

    // Testbench signals
    reg clk;
    wire [15:0] phase;
    wire [15:0] data;

    // Instantiate the DDS module
    dds_compiler_0 uut (
      .aclk(clk),                                // input wire aclk
      .m_axis_data_tvalid(),    // output wire m_axis_data_tvalid
      .m_axis_data_tdata(data),      // output wire [15 : 0] m_axis_data_tdata
      .m_axis_phase_tvalid(),  // output wire m_axis_phase_tvalid
      .m_axis_phase_tdata(phase)    // output wire [15 : 0] m_axis_phase_tdata
    );

    // Clock generation
    always @(clk) #1 clk <= ~clk;

    // Test sequence
    initial begin
        clk = 1;
        #60000;
        $stop;
    end


endmodule
```



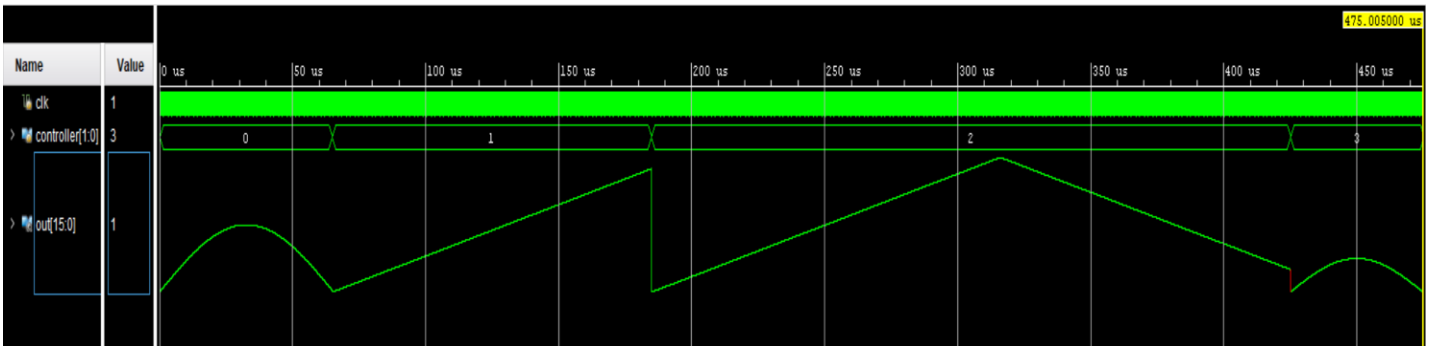
(d

```
%% Q1_d
clc; clear; close all;

mem_len = 50000;

x = linspace(0, 2*pi, mem_len);
q = quantizer([16,14]);

y_sin = num2bin(q, sin(x));

sin = fopen('sin.txt', 'w');

for i = 1 : mem_len
    fprintf(sin, '%s', y_sin(i,:));
    if (i ~= mem_len)
        fprintf(sin, '\n');
    end
end
```

```
`timescale 1ns/1ps
module Q1_d_tb;

    reg clk;
    reg [15:0] datas [50000-1:0];
    reg [15:0] out;
    integer i;

    initial begin
        $readmemb("sin.txt",datas);
        clk <= 1;
        for (i = 0; i<50000 ; i=i+1) begin
            #5
            out <= datas[i];
        end
        #5
        $stop;

    end

    always @(clk) #1 clk <= ~clk;

endmodule
```



e) در این قسمت صرفا قسمت های قبلی را ادقام کردم.



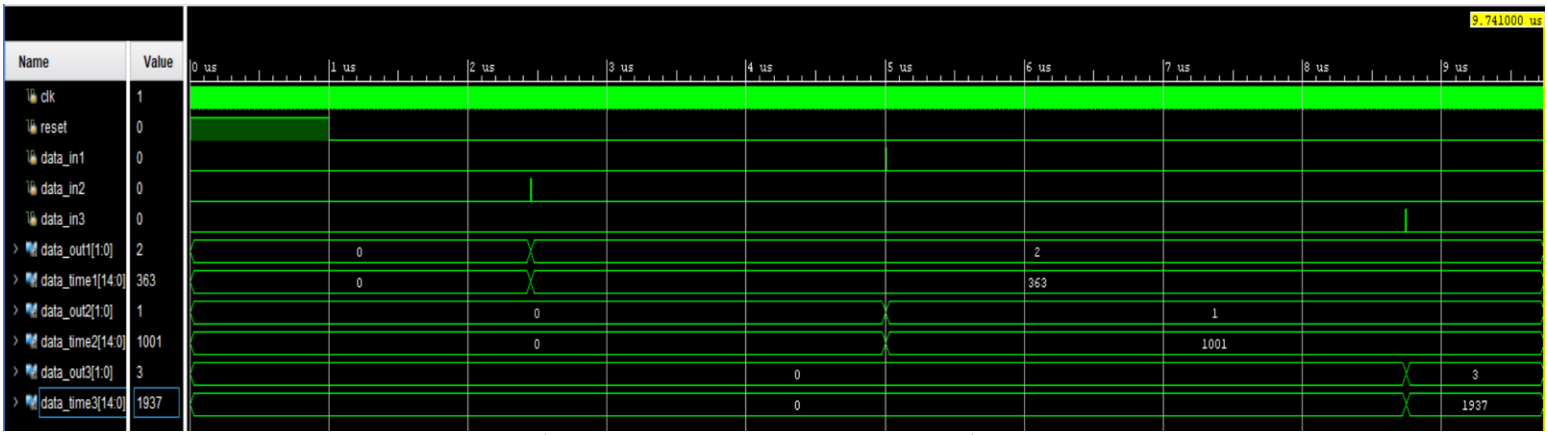
سوال 2) این 2 ماژول یک عیب دارد از جمله اینکه چون ماژول اول عمق خروجی اش برابر ورودی است ولی برای دومی 2 برابر این میزان است که اشتباه است و ممکن است اگر مقدار حاصل ضرب در ان جا نشود و اور فلو کند که در این صورت بیت های بالایی ان دور ریخته میشوند که مقدار اشتباهی را حاصل میدهد.

```
module Q2_tb;

    reg [7 : 0] in1, in2;
    reg inValid, clk, en, reset;
    wire [7 : 0] out1;
    wire [15 : 0] out2;
    wire outValid1, outValid2;

    always @(clk) #5 clk <= ~clk;

    Mult1 uut1 (
        .in1(in1),
        .in2(in2),
        .inValid(inValid),
        .clk(clk),
        .en(en),
        .reset(reset),
        .out(out1),
        .outValid(outValid1)
    );

    Mult2 uut2 (
        .in1(in1),
        .in2(in2),
        .inValid(inValid),
        .clk(clk),
        .en(en),
        .reset(reset),
        .out(out2),
        .outValid(outValid2)
    );

    initial begin
        clk = 1;
        inValid = 1;
        en = 1;
        reset = 1;

        repeat (10) begin
            in1 = $random;
            in2 = $random;
            #40
            $display("\nmult1: %0d * %0d = %0d",in1,in2,out1);
            $display("mult2: %0d * %0d = %0d",in1,in2,out2);
        end
        $display("\n");

        #40
        $stop;
    end

endmodule
```

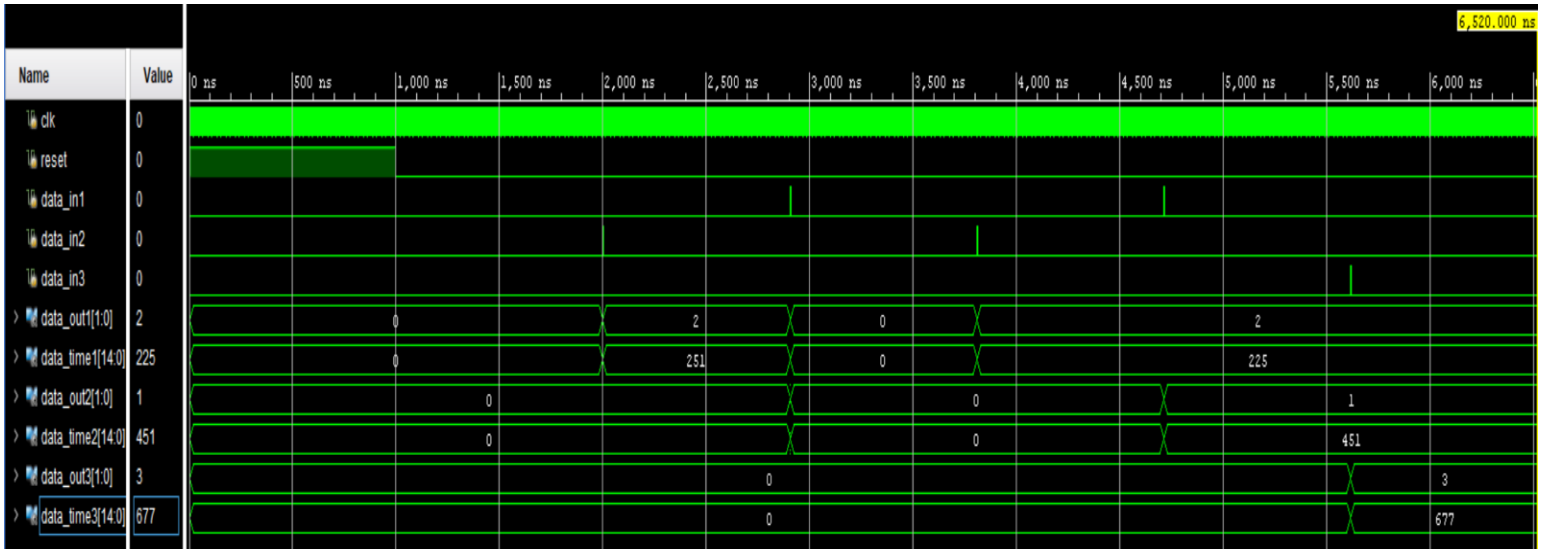
```
VSIM 27> run -all
#
# mult1: 36 * 129 = 36
# mult2: 36 * 129 = 4644
#
# mult1: 9 * 99 = 123
# mult2: 9 * 99 = 891
#
# mult1: 13 * 141 = 41
# mult2: 13 * 141 = 1833
#
# mult1: 101 * 18 = 26
# mult2: 101 * 18 = 1818
#
# mult1: 1 * 13 = 13
# mult2: 1 * 13 = 13
#
# mult1: 118 * 61 = 30
# mult2: 118 * 61 = 7198
#
# mult1: 237 * 140 = 156
# mult2: 237 * 140 = 33180
#
# mult1: 249 * 198 = 150
# mult2: 249 * 198 = 49302
#
# mult1: 197 * 170 = 210
# mult2: 197 * 170 = 33490
#
# mult1: 229 * 119 = 115
# mult2: 229 * 119 = 27251
#
#
# ** Note: $stop    : C:/Users/gilan/OneDrive/Desktop/Q2-tb.v(49)
#    Time: 440 ns  Iteration: 0  Instance: /Q2_tb
```

سوال 3)

a,b) در این قسمت صرفا یک رجیستر برای پر بودن هر خروجی تعیین کردم و گفتم اگه ورودی امد به ترتیب در این خانه ها بگزار و سپس در قسمت بعد ان را تست کردم.



c) در این قسمت هم صرفا گفتم که 2 ورودی اول را در نظر نگیر و دولاره از اول بشمر.



سوال 4) این سوال هم نکته خاصی نداشت صرفا انها را پیاده سازی کردم و خروجی ها را در فایل های جدا ذخیره کردم.

```
module Q4 (
        input wire clk,
        input wire reset,
        input wire [31:0] data_in,
        output reg ram_en,
        output reg [15:0] ram_data,
        output reg [7:0] ram_address,
        output reg error
    );

    always @(posedge clk) begin
        if (!reset) begin
            ram_en <= 0;
            ram_address <= 0;
            ram_data <= 0;
            error <= 0;
        end
        else begin
            if (data_in[3:0] == 4'he) begin
                ram_address <= data_in[11:4];
                ram_data <= data_in[27:12];
                ram_en <= 1;
                error <= 0;
            end
            else begin
                ram_address <= 0;
                ram_data <= 0;
                ram_en <= 0;
                error <= 1;
            end
        end
    end

endmodule
```

```
`timescale 1ns/1ps
module Q4_tb;

    reg clk;
    reg reset;
    reg [31:0] data_in;
    wire ram_en;
    wire [15:0] ram_data;
    wire [7:0] ram_address;
    wire error;

    integer error_file, address_file, data_file;

    Q4 uut (
        .clk(clk),
        .reset(reset),
        .data_in(data_in),
        .ram_en(ram_en),
        .ram_address(ram_address),
        .ram_data(ram_data),
        .error(error)
    );

    always @(clk) #5 clk <= ~clk;

    initial begin
        data_in = 0;
        reset = 1;
        clk = 1;

        error_file = $fopen("error.txt","w");
        address_file = $fopen("address.txt", "w");
        data_file = $fopen("data.txt", "w");

        #10
        repeat (20) begin
            data_in = $random();
            data_in = {data_in[31:4], ($random%32 > 5)? 4'he : 4'ha};
            #10;
            if (ram_en) begin
                $fwrite(error_file,"Error : %b\n", error);
                $fwrite(address_file,"Address : %b\n", ram_address);
                $fwrite(data_file,"Data : %b\n", ram_data);
            end
            else begin
                $fwrite(error_file,"Error : %b\n", error);
                $fwrite(address_file,"\n");
                $fwrite(data_file,"\n");
            end
        end

        $fclose();
        $stop;
    end

endmodule
```