Lab 4: parallel-load register and BCD counter

3.1 Design a 16-bit parallel-load register (20 points)

Verilog Source:

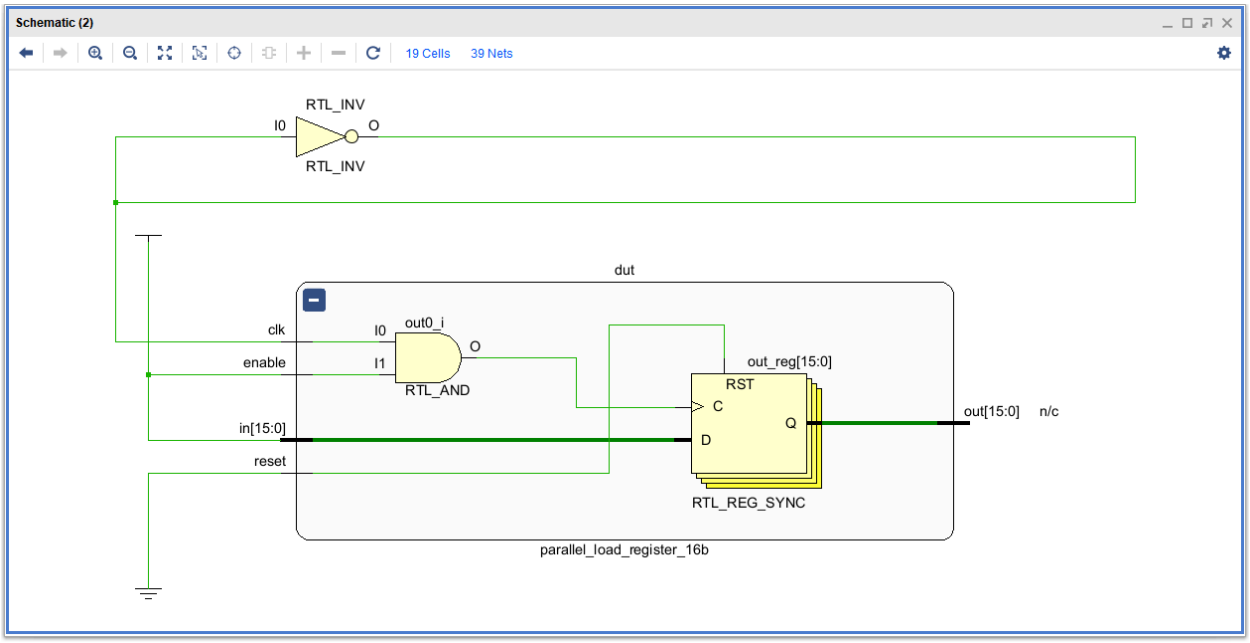
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2023 06:12:01 PM
// Design Name:
// Module Name: parallel_load_register_16b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module parallel_load_register_16b(out, reset, clk, enable, in);

    output reg [15:0] out;
    input [15:0] in;
    input reset, clk, enable;

    always @(posedge clk & enable) begin
        if (reset) begin
            out = 0;
        end
        else begin
            out = in;
        end
    end
endmodule
```

Testbench Source:

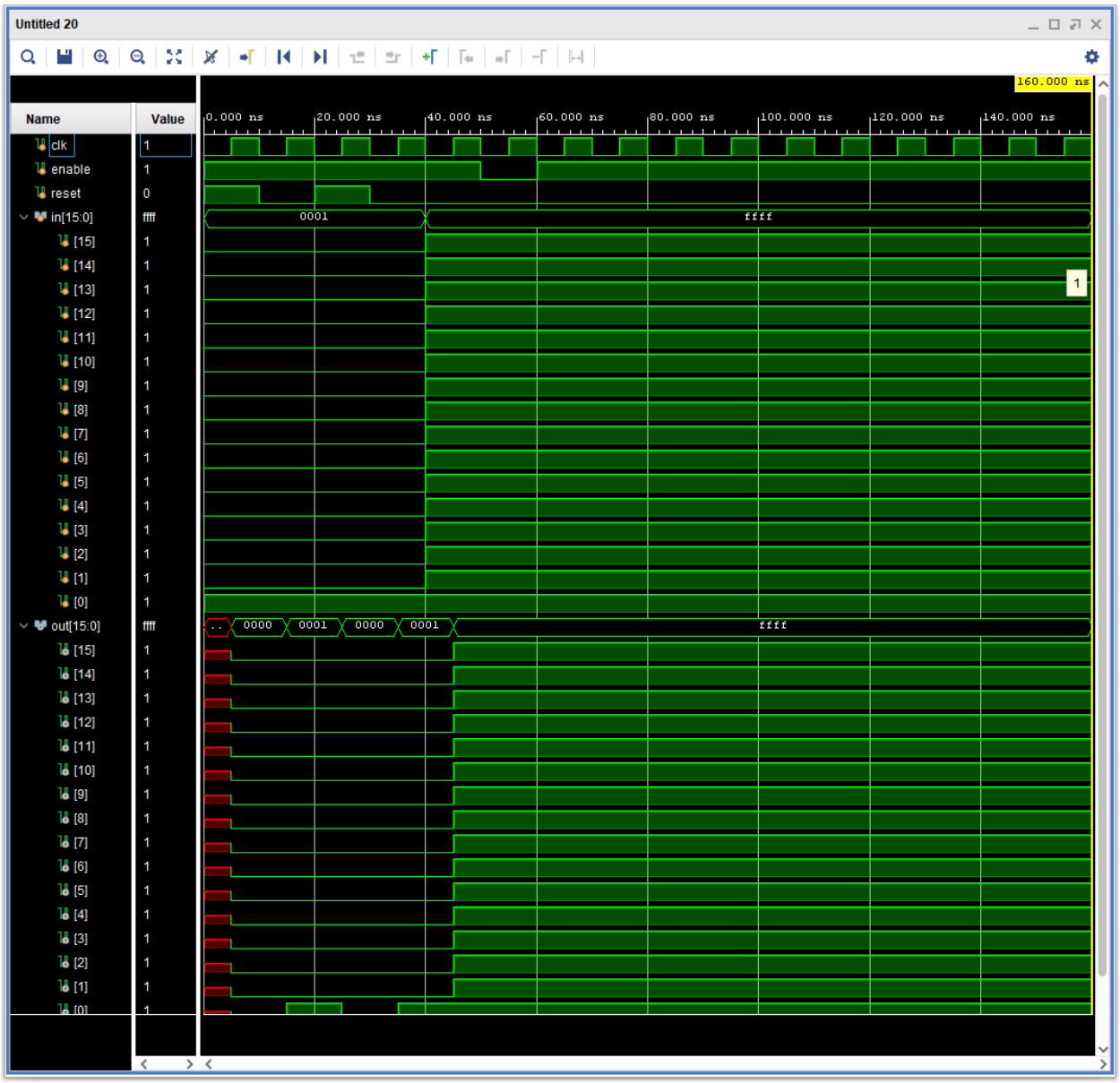
```
// Testbench for parallel_load_register_16b module

`timescale 1ns / 1ps

module parallel_load_register_16b_tb;

  // Inputs
  reg clk = 0;
  reg enable;
  reg reset;
  reg [15:0] in;

  // Outputs
  wire [15:0] out;
```

```
  // Instantiate the module
  parallel_load_register_16b dut (
    .out(out),
    .reset(reset),
    .clk(clk),
    .enable(enable),
    .in(in)
  );

  // Clock generator
  always #5 clk = ~clk;

  // Stimulus
  initial begin
    // Initialize inputs
    enable = 1;
    reset = 1;
    in = 16'b0000_0000_0000_0001;

    // Reset sequence
    #10 reset = 0;
    #10 reset = 1;
    #10 reset = 0;

    // Load input value
    #10 in = 16'b1111_1111_1111_1111;
    #10 enable = 0;
    #10 enable = 1;

    // Wait for a few more clock cycles to observe the outputs
    #100 $finish;
  end

  // Monitor
  always @(posedge clk) begin
    $display("out = %d", out);
  end

endmodule
```

RTL Schematic:



Simulation Result:



3.2 Design a 4-bit BCD counter (25 points)

Verilog Source:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2023 06:12:01 PM
// Design Name:
// Module Name: BCD_counter_4b
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
```

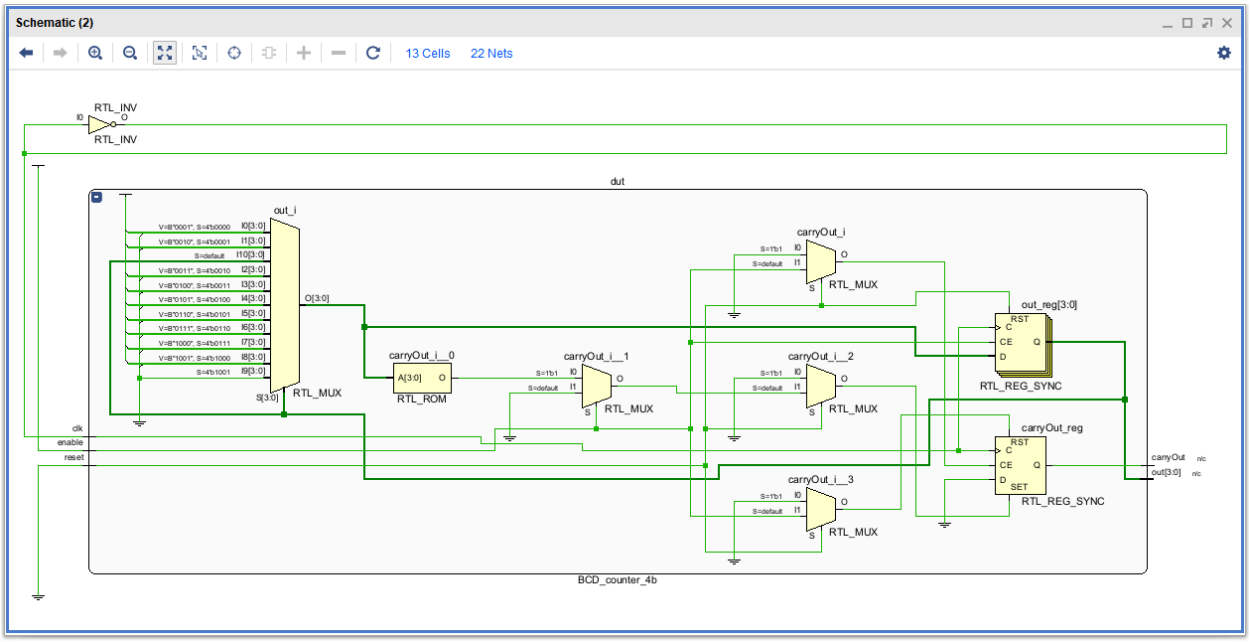
```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module BCD_counter_4b(out, carryOut, clk, enable, reset);

    output reg [3:0] out;
    output reg carryOut;
    input clk, enable, reset;

    always @(posedge clk) begin

        if (reset) begin
            out = 0;
        end
        else begin
            if (enable) begin
                case (out)
                    4'b0000: out = 4'b0001;
                    4'b0001: out = 4'b0010;
                    4'b0010: out = 4'b0011;
                    4'b0011: out = 4'b0100;
                    4'b0100: out = 4'b0101;
                    4'b0101: out = 4'b0110;
                    4'b0110: out = 4'b0111;
                    4'b0111: out = 4'b1000;
                    4'b1000: out = 4'b1001;
                    4'b1001: out = 4'b0000;
                endcase
                if (out == 4'b1001) begin
                    carryOut = 1'b1;
                end
                else begin
                    carryOut = 1'b0;
                end
            end
        end
    end
endmodule
```

Testbench Source:

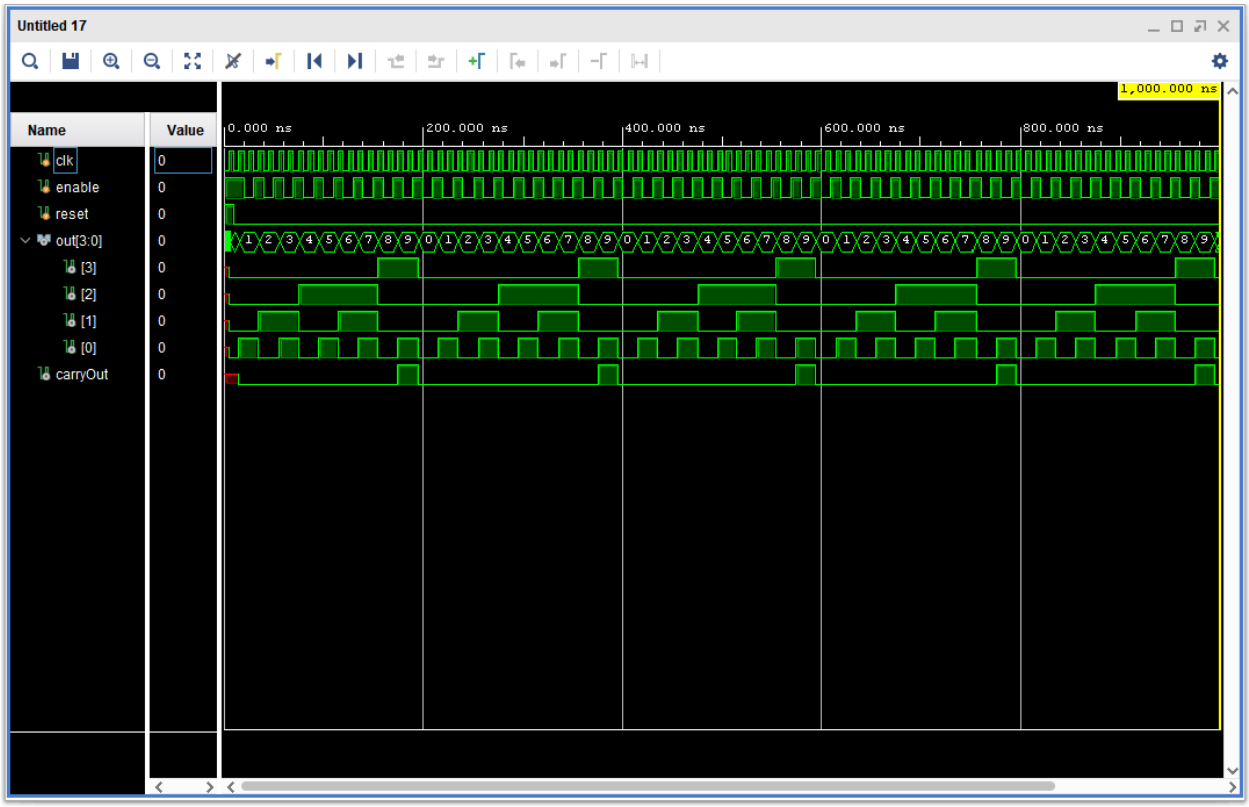
```
// Testbench for BCD_counter_4b module
`timescale 1ns / 1ps

module BCD_counter_4b_tb;

    // Inputs
    reg clk = 0;
    reg enable;
    reg reset;

    // Outputs
    wire [3:0] out;
    wire carryOut;

    // Instantiate the module
    BCD_counter_4b dut(
```

```
        .out(out),
        .carryOut(carryOut),
        .clk(clk),
        .enable(enable),
        .reset(reset)
    );

    // Generate clock signal
    always #5 clk = ~clk;

    // Stimulus
    initial begin
        // Initialize inputs
        reset = 1;
        enable = 1;

        // Reset the counter
        #10 reset = 0;

        // Test counting up from 0 to 9
        for (integer i = 0; i < 100; i = i + 1) begin
            #10 enable = 0;
            #10 enable = 1;
        end

        #10 $finish;
    end

    // Monitor output
    always @(posedge clk) begin
        $display("out: %d carryOut: %d", out, carryOut);
    end

endmodule
```

RTL Schematic:



Simulation Result:



3.3 Design a 4-digit BCD counter (30 points)

Verilog Source:

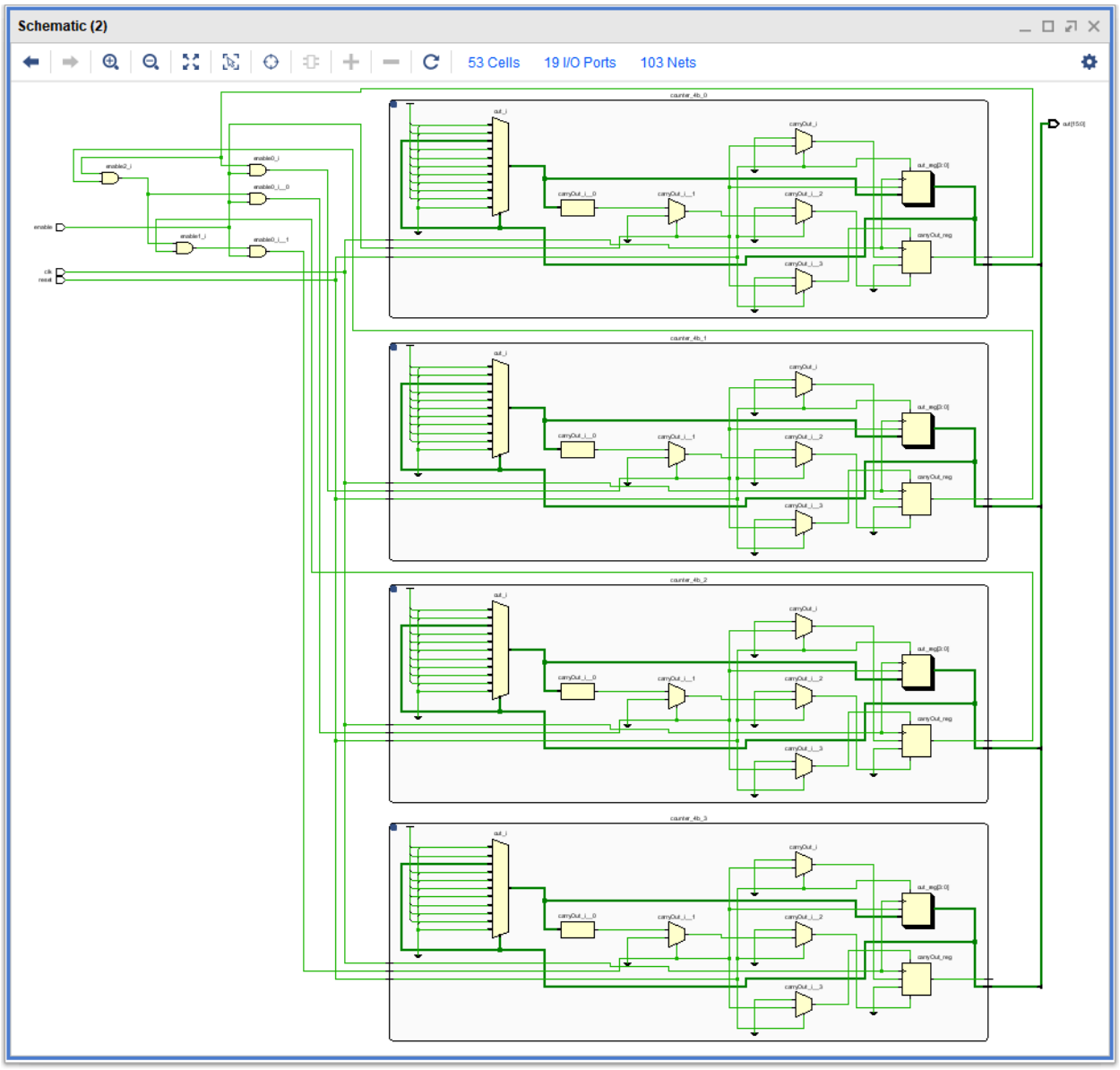
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2023 06:12:01 PM
// Design Name:
// Module Name: BCD_counter_4d
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module BCD_counter_4d(out, clk, enable, reset);

    output reg [15:0] out = 4'h0000;
    input clk, enable, reset;

    wire [3:0] out0_3, out4_7, out8_11, out12_15;
    wire CO_0, CO_1, CO_2, CO_3;

    // BCD_counter             (       out, carryOut, clk,                     enable, reset);
```

```
    BCD_counter_4b counter_4b_0(  out0_3,     CO_0, clk,                     enable, reset);
    BCD_counter_4b counter_4b_1(  out4_7,     CO_1, clk,               CO_0 & enable, reset);
    BCD_counter_4b counter_4b_2( out8_11,     CO_2, clk,        CO_0 & CO_1 & enable, reset);
    BCD_counter_4b counter_4b_3(out12_15,     CO_3, clk, CO_0 & CO_1 & CO_2 & enable, reset);

    always @(*) begin
        out = {out12_15, out8_11, out4_7, out0_3};
    end
endmodule
```

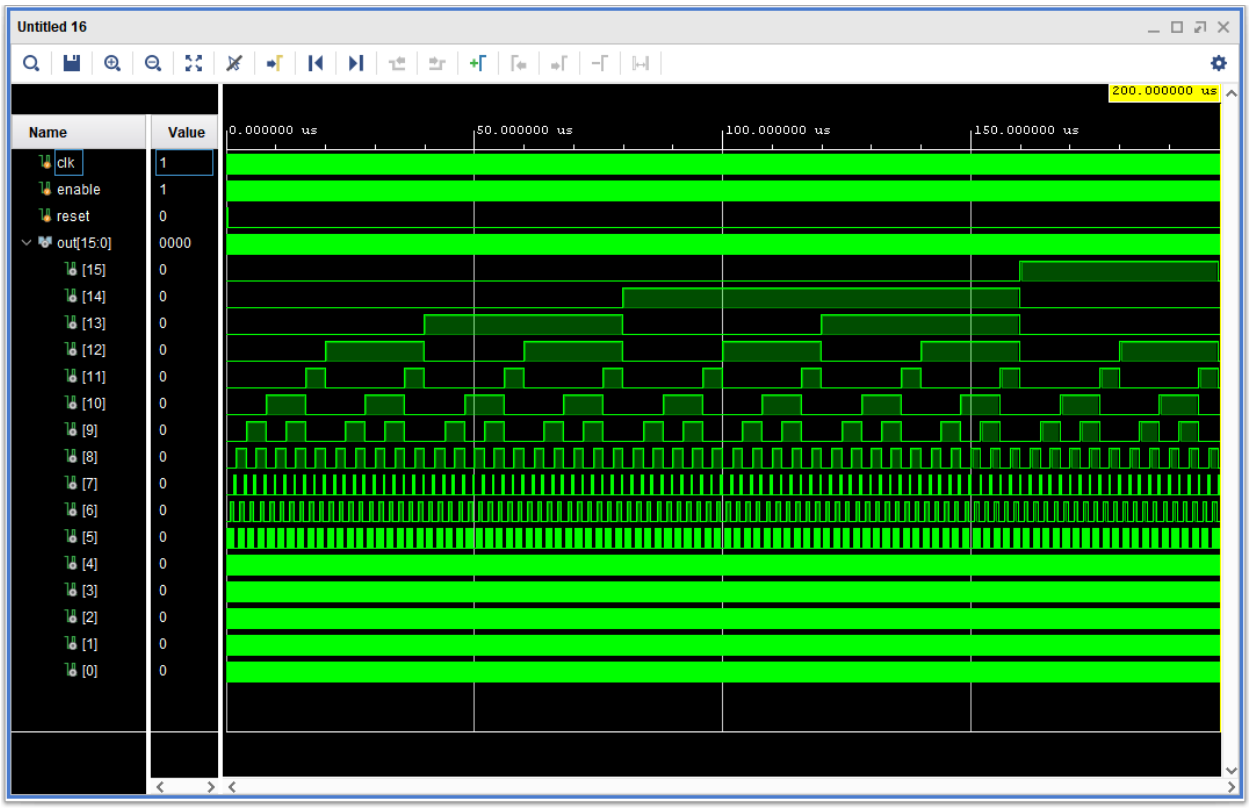
Testbench Source:

```
// Testbench for BCD_counter_4d module

`timescale 1ns / 1ps

module test_BCD_counter_4d;

    // Inputs
    reg clk = 0;
    reg enable;
    reg reset;

    // Outputs
    wire [15:0] out;

    // Instantiate the module
    BCD_counter_4d dut (
        .out(out),
        .clk(clk),
        .enable(enable),
        .reset(reset)
    );

    // Clock generator
    always #5 clk = ~clk;

    integer i;

    // Stimulus
    initial begin
        // Initialize inputs
        enable = 1;
        reset = 1;

        // Reset sequence
        #10 reset = 0;

        // Count sequence
        for (i = 0; i < 9999; i = i + 1) begin
            #10 enable = 0;
            #10 enable = 1;
        end

        // Wait for a few more clock cycles to observe the outputs
        #10 $finish;
    end

    // Monitor
    always @(posedge clk) begin
        $display("out = %d", out);
    end

endmodule
```

RTL Schematic:



Simulation result:



3.4 Implement the 4-digit BCD counter on FPGA and demo the counter on the 4 seven-segment displays (25 points)