PROJECT REPORT

REAL-TIME CLOCK (RTC) USING VERILOG HDL

Submitted by:
SHEFIN MUHAMMED M S

Under the Guidance of:
MR.MONISH & MS.VIMALA

SION SEMICONDUCTOR

October 2025

Contents

1. Aim

The aim of this project is to design and implement a **Real-Time Clock (RTC)** using Verilog HDL that can display the current time in hours, minutes, and seconds in Seven Segmented Display in format with proper rollover logic in a 24-hour format.

2. Objectives

- To design an RTC circuit using Verilog HDL.
- To implement counters for seconds, minutes, and hours.
- To provide automatic rollover after 23:59:59 hour.
- To display using Seven Segmented Display
- To verify the design functionality using ModelSim simulation.
- To synthesize and implement the design using Xilinx Vivado.

3. Tools Used

- **Software Tools:**
  - ModelSim – for simulation and waveform verification.
  - Xilinx Vivado – for synthesis and RTL implementation.

4. Methodology

1. The RTC is designed using Verilog HDL with three counters for hours, minutes, and seconds.
2. Each counter increments based on the clock signal and rolls over when it reaches its limit 60 for seconds and 60 for minutes, 24 for hours.
3. Display using Seven Segmented Display
4. The design includes control logic for resetting and updating the clock.
5. Simulation is performed in ModelSim to verify correct time incrementation and rollover behavior.

6. The synthesized design is analyzed in Xilinx Vivado using RTL and Technology Schematic views.

5. Verilog Code and TestBench

5.1. Verilog Code

```
module RTC_counter #(parameter N = 9)(input clk,rst,clear,en,
   output reg [3:0]count,output next);

  always@(posedge clk)begin
    if(rst|clear)
      count<=0;
    else if(en)begin
      if(N==count)
        count<=0;
    else
        count<=count+1;
    end
  end
  assign next=(N==count)&&en;
endmodule

module RTC(input clk,rst,output [3:0] hrm,hrl,minm,minl,secm,secl
   ,output [6:0]sec_l,sec_m,min_m,min_l,hr_m,hr_l);

wire next_secl,next_secm,next_minl,next_minm,next_hrl,next_hrm,
   reset_clk;
  wire clear_rst=(hrm==2&&hrl==3 && minm==5&&minl==9 && secm==5
     && secl ==9); //CHECKING THE CONDITION FOR RESET THE CLK

  //seconds
  RTC_counter #(9) sec_lsb (.clk(clk),.rst(rst),.clear(clear_rst)
     ,.en(1'b1),.count(secl),.next(next_secl));

  RTC_counter #(5) sec_msb (.clk(clk),.rst(rst),.clear(clear_rst)
     ,.en(next_secl),.count(secm),.next(next_secm));

  //min
```

```
27    RTC_counter #(9) min_lsb (.clk(clk),.rst(rst),.clear(clear_rst)
        ,.en(next_secm),.count(minl),.next(next_minl));
28
29    RTC_counter #(5) min_msb (.clk(clk),.rst(rst),.clear(clear_rst)
        ,.en(next_minl),.count(minm),.next(next_minm));
30
31    //hours
32
33    RTC_counter #(9) hr_lsb (.clk(clk),.rst(rst),.clear(clear_rst)
        ,.en(next_minm),.count(hrl),.next(next_hrl));
34
35    RTC_counter #(2) hr_msb (.clk(clk),.rst(rst),.clear(clear_rst)
        ,.en(next_hrl),.count(hrm),.next(next_hrm));
36
37    //MODULE INSTANTIATION FOR DISPLAYING THE OUTPUT USING 7
        SEGMENTED DISPLAY
38    seven_segmented_dis s1(.bcd(secl),.sev_dis(sec_l));
39    seven_segmented_dis sm(.bcd(secm),.sev_dis(sec_m));
40    seven_segmented_dis mm(.bcd(minm),.sev_dis(min_m));
41    seven_segmented_dis ml(.bcd(minl),.sev_dis(min_l));
42    seven_segmented_dis hm(.bcd(hrm),.sev_dis(hr_m));
43    seven_segmented_dis hl(.bcd(hrl),.sev_dis(hr_l));
44
45
46  endmodule
47
48
49  //CODE FOR SEVEN SEGMENTED DISPLAY
50  module seven_segmented_dis(input [3:0]bcd,output reg [6:0]sev_dis
      );
51    always@(*)begin
52      case(bcd)
53              4'd0: sev_dis = 7'b1111110;
54              4'd1: sev_dis = 7'b0110000;
55              4'd2: sev_dis = 7'b1101101;
56              4'd3: sev_dis = 7'b1111001;
57              4'd4: sev_dis = 7'b0110011;
58              4'd5: sev_dis = 7'b1011011;
59              4'd6: sev_dis = 7'b1011111;
```

```
            4'd7: sev_dis = 7'b1110000;
            4'd8: sev_dis = 7'b1111111;
            4'd9: sev_dis = 7'b1111011;
            default: sev_dis = 7'b0000000;
        endcase

  end
endmodule
```

5.2. Testbench code

```
//TESTBENCH CODE FOR RTC
module RTC_counter_tb;
  reg clk,rst;
  wire [3:0] hrm,hrl,minm,minl,secm,secl;
  wire[6:0]sec_l,sec_m,min_m,min_l,hr_m,hr_l;

  RTC dut(.clk(clk),.rst(rst),.hrm(hrm),
    .hrl(hrl),
    .minm(minm),
    .minl(minl),
    .secm(secm),
    .secl(secl),
     .sec_l(sec_l),
    .sec_m(sec_m),
    .min_m(min_m),
    .min_l(min_l),
    .hr_m(hr_m),
    .hr_l(hr_l)
         );

  initial clk=0;
  always#5 clk= ~clk;

  initial begin
    rst=1;#10
    rst=0;

    #2000000;
```

```
29
30      //rst=1;#10;
31      //rst=0;
32      //#100000;
33
34      $stop;
35    end
36      // Monitor outputs
37    initial begin
38      $display("Time|HRMHRL   | MINMMINL | SECMSECL----seven
           segmented display out");
39      $monitor("%0t | %0d%0d   %0d%0d      %0d%0d || seven display
           out = %b | %b | %b | %b | %b | %b",$time, hrm, hrl, minm,
           minl, secm, secl,hr_m,hr_l,min_m,min_l,sec_m,sec_l);
40    end
41
42  endmodule
```

6. Results

The Verilog-based RTC successfully counts time in hours, minutes, and seconds format. Simulation results confirm accurate rollover and timekeeping.

6.1. Simulated Result

```
# 199655 | 05   32      45 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 0110011 | 1011011
# 199665 | 05   32      46 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 0110011 | 1011111
# 199675 | 05   32      47 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 0110011 | 1110000
# 199685 | 05   32      48 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 0110011 | 1111111
# 199695 | 05   32      49 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 0110011 | 1111011
# 199705 | 05   32      50 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1111110
# 199715 | 05   32      51 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 0110000
# 199725 | 05   32      52 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1101101
# 199735 | 05   32      53 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1111001
# 199745 | 05   32      54 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 0110011
# 199755 | 05   32      55 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1011011
# 199765 | 05   32      56 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1011111
# 199775 | 05   32      57 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1110000
# 199785 | 05   32      58 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1111111
# 199795 | 05   32      59 || seven display out = 1111110 | 1011011 | 1111001 | 1101101 | 1011011 | 1111011
# 199805 | 05   33      00 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1111110
# 199815 | 05   33      01 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 0110000
# 199825 | 05   33      02 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1101101
# 199835 | 05   33      03 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1111001
# 199845 | 05   33      04 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 0110011
# 199855 | 05   33      05 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1011011
# 199865 | 05   33      06 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1011111
# 199875 | 05   33      07 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1110000
# 199885 | 05   33      08 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1111111
# 199895 | 05   33      09 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1111110 | 1111011
# 199905 | 05   33      10 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1111110
# 199915 | 05   33      11 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 0110000
# 199925 | 05   33      12 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1101101
# 199935 | 05   33      13 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1111001
# 199945 | 05   33      14 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 0110011
# 199955 | 05   33      15 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1011011
# 199965 | 05   33      16 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1011111
# 199975 | 05   33      17 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1110000
# 199985 | 05   33      18 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1111111
# 199995 | 05   33      19 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 0110000 | 1111011
# 200005 | 05   33      20 || seven display out = 1111110 | 1011011 | 1111001 | 1111001 | 1101101 | 1111110
```

Figure 1: Simulated Result

```
# 863905 | 23   59      50 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1111110
# 863915 | 23   59      51 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 0110000
# 863925 | 23   59      52 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1101101
# 863935 | 23   59      53 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1111001
# 863945 | 23   59      54 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 0110011
# 863955 | 23   59      55 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1011011
# 863965 | 23   59      56 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1011111
# 863975 | 23   59      57 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1110000
# 863985 | 23   59      58 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1111111
# 863995 | 23   59      59 || seven display out = 1101101 | 1111001 | 1011011 | 1111011 | 1011011 | 1111011
# 864005 | 00   00      00 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1111110
# 864015 | 00   00      01 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 0110000
# 864025 | 00   00      02 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1101101
# 864035 | 00   00      03 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1111001
# 864045 | 00   00      04 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 0110011
# 864055 | 00   00      05 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1011011
# 864065 | 00   00      06 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1011111
# 864075 | 00   00      07 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1110000
# 864085 | 00   00      08 || seven display out = 1111110 | 1111110 | 1111110 | 1111110 | 1111110 | 1111111
```

Figure 2: Roll Over after 23:59:59

6.2. Waveform

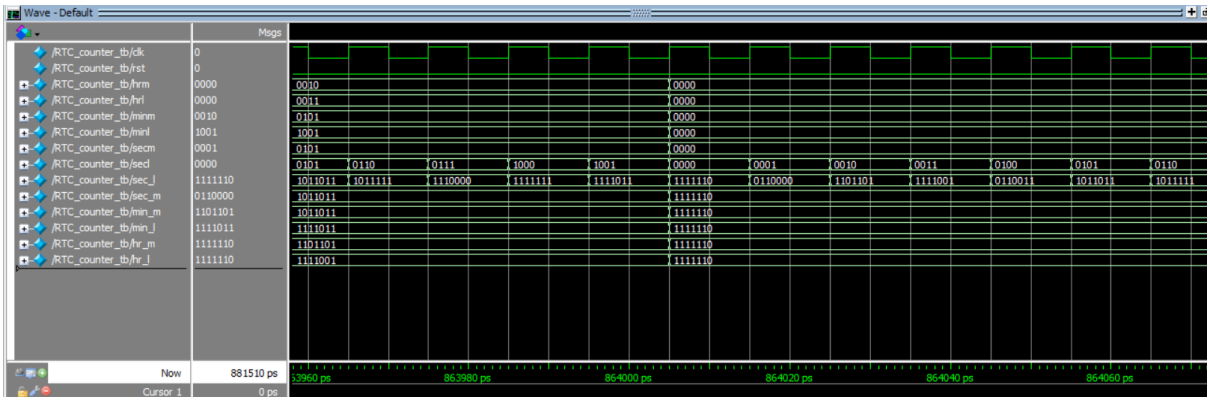

Figure 3: Waveform

6.3. Elaborated Design

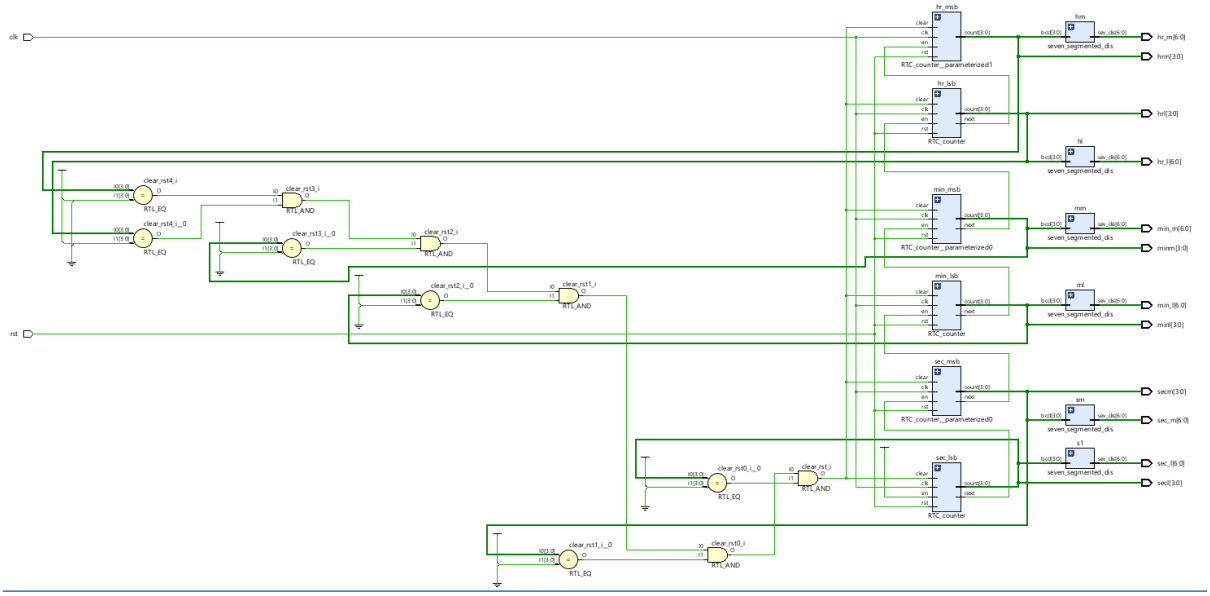

Figure 4: Elaborated Design using Vivado

7. Conclusion

The project demonstrates the design and verification of a Real-Time Clock using Verilog HDL. The system efficiently counts and displays real-time values, proving useful for various timing-based digital systems.