REAL TIME CLOCK
Mini-Project

Name: Ganesh Prasad B K

ID Number: ganeshp@sionsemi.com

Date: 22 Aug 2020

Block Diagram:

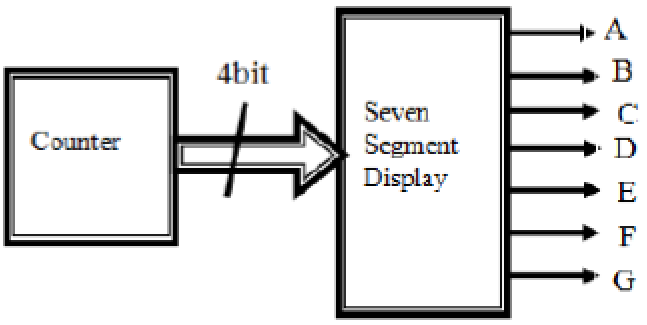


Figure 1 Seven Segment Display of Real Time Digital Clock

The basic counter is count the hours, minutes and seconds and set as **00:00:00** when it will reach **23:59:59**.

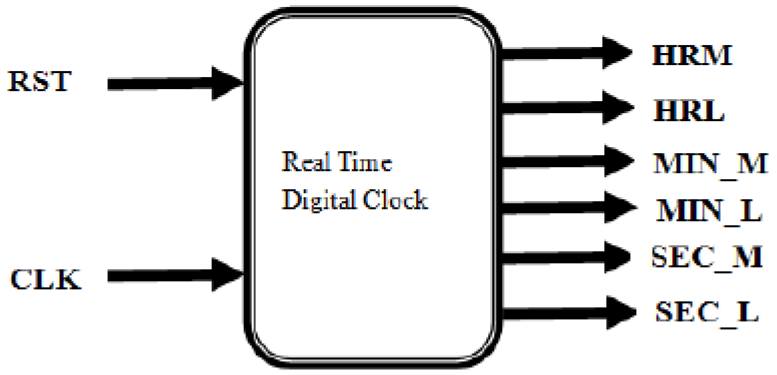


Figure 2 Block diagram of Real Rime Digital Clock

Note:
To open the Xilinx project directly, double click & open Real_Time_Clock.xpr
Path: ~\Xilinx_implementation\Real_Time_Clock\Real_Time_Clock.xpr

Real Time Clock

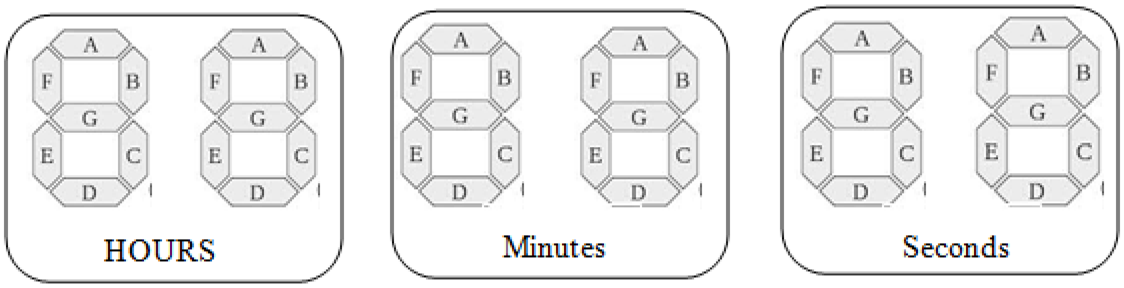
Introduction:

Real Time Clock (RTC) is an important device for good relation with world. The first application is to display the real time on 24h basis and it can be easily converted to accommodate a 12h clock as well. The RTC can display in hours, minutes and seconds. This design is more involved in the traffic controller design. The display system needs to be populated with all the six, seven-segment LEDs for the real time clock applications.
The main applications of RTC are

- Real Time Display
- Stop Watch
- Industrial timer
- Photographic timer
- Medical Application using three alarm setting

Design Process of RTC:

The basic digital clock contained hours, minutes and seconds. Each one has MSB and LSB.



The above seven segments are illustrated the "00-00-00" values of hours, HRM (hours of MSB value) and HRL (Hours of LSB value). Next will declared minutes as MIN_M and MIN_L. The seconds are declared as SEC_M and SEC_L.

The basic digital block contain two designs,

- Counter
- Seven Segmentation

These are the main building blocks of digital clock. Counter will count the number in decimal as 23:59:59. This is the 24h clock operation and it can easily convert to accommodate a 12h clock.

Real_Time_Digital_clock.v->Top Module

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21.08.2020 10:15:59
// Design Name:
// Module Name: Real_Time_Digital_Clock
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Real_Time_Digital_Clock(
    output [6:0] HRM,
    output [6:0] HRL,
    output [6:0] MIN_M,
    output [6:0] MIN_L,
    output [6:0] SEC_M,
    output [6:0] SEC_L,
    input CLK,
    input RST,
    input format_ctrl
    );

wire [3:0]hrM, hrL, minM, minL, secM, secL;

wire [7:0]format;

wire ovhrM, ovhrL, ovminM, ovminL, ovsecM, ovsecL;

//format_ctrl=1:24-hr format clock
//format_ctrl=0:12-hr format clock
assign format = format_ctrl? 8'b0011_0100 : 8'b0010_0011;

//clock counting mechanism
bcd_counter hourM_cntr(.count(hrM), .ov(ovhrM), .clk(ovhrL), .mode(format[7:4]),
.rset(RST));
bcd_counter hourL_cntr(.count(hrL), .ov(ovhrL), .clk(ovminM), .mode(format[3:0]),
.rset(RST));
bcd_counter minuteM_cntr(.count(minM), .ov(ovminM), .clk(ovminL), .mode(4'd6),
.rset(RST));
bcd_counter minuteL_cntr(.count(minL), .ov(ovminL), .clk(ovsecM), .mode(4'd10),
.rset(RST));
bcd_counter secondM_cntr(.count(secM), .ov(ovsecM), .clk(ovsecL), .mode(4'd6),
.rset(RST));
bcd_counter secondL_cntr(.count(secL), .ov(ovsecL), .clk(CLK), .mode(4'd10), .rset(RST));

//connecting clock counting mechanism to &-segment display
bcd_to_7segment_disp hourM(.out(HRM), .in(hrM));
bcd_to_7segment_disp hourL(.out(HRL), .in(hrL));
bcd_to_7segment_disp minuteM(.out(MIN_M), .in(minM));
bcd_to_7segment_disp minuteL(.out(MIN_L), .in(minL));
bcd_to_7segment_disp secondM(.out(SEC_M), .in(secM));
bcd_to_7segment_disp secondL(.out(SEC_L), .in(secL));

initial
```

```verilog
//to dispaly time in transcript window in HH:MM:SS format
$monitor("%d%d : %d%d : %d%d",hrM, hrL, minM, minL, secM, secL);

endmodule

```

bcd_counter

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21.08.2020 10:15:59
// Design Name:
// Module Name: bcd_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module bcd_counter(
    output [3:0] count,
    output ov,
    input [3:0] mode,
    input clk,
    input rset
    );

reg [3:0]count_temp = 4'd0;
reg ov_temp = 0;

assign count = count_temp;
assign ov = ov_temp;

always@(posedge clk)
begin
    if(rset)
    begin
        count_temp <= 4'd0;
        ov_temp <= 0;
    end
    else
    begin
        case(mode)
        4'd2:begin      //mod-2 counter
            if(count_temp < 'd1 )
            begin
                count_temp <= count_temp + 4'd1;
                ov_temp <= 0;
            end
            else
            begin
                count_temp <= 4'd0;
                ov_temp <= 1;
            end
        end
        4'd3:begin      //mod-3 counter
            if(count_temp < 'd2 )
            begin
                count_temp <= count_temp + 4'd1;
                ov_temp <= 0;
            end
            else
            begin
```

```
                    count_temp <= 4'd0;
                    ov_temp <= 1;
                end
            end
            4'd4:begin      //mod-4 counter
                if(count_temp < 'd3 )
                begin
                    count_temp <= count_temp + 4'd1;
                    ov_temp <= 0;
                end
                else
                begin
                    count_temp <= 4'd0;
                    ov_temp <= 1;
                end
            end
            4'd6:begin      //mod-6 counter
                if(count_temp < 'd5 )
                begin
                    count_temp <= count_temp + 4'd1;
                    ov_temp <= 0;
                end
                else
                begin
                    count_temp <= 4'd0;
                    ov_temp <= 1;
                end
            end
            4'd10:begin     //mod-10 counter
                if(count_temp < 'd9 )
                begin
                    count_temp <= count_temp + 4'd1;
                    ov_temp <= 0;
                end
                else
                begin
                    count_temp <= 4'd0;
                    ov_temp <= 1;
                end
            end
            default:begin
                count_temp <= 0;
                ov_temp <= 0;
                end
            endcase
        end
    end

    endmodule

```

bcd_to_7segment_disp.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21.08.2020 10:15:59
// Design Name:
// Module Name: bcd_to_7segment_disp
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module bcd_to_7segment_disp(
    output [6:0] out,
    input [3:0] in
    );

reg [6:0]y = 7'b1111110;
assign out = y;

always@(in)
begin
    case(in)
    4'd0:   y <= 7'b1111110;// 7Eh = 126d
    4'd1:   y <= 7'b0110000;// 30h = 48d
    4'd2:   y <= 7'b1101101;// 6Dh = 109d
    4'd3:   y <= 7'b1111001;// 79h = 121d
    4'd4:   y <= 7'b0110011;// 33h = 51d
    4'd5:   y <= 7'b1011011;// 5Bh = 91d
    4'd6:   y <= 7'b1011111;// 5Fh = 95d
    4'd7:   y <= 7'b1110000;// 70h = 112d
    4'd8:   y <= 7'b1111111;// 7Fh = 127d
    4'd9:   y <= 7'b1111011;// 7Bh = 123d
    default:y <= 7'b1111110;// 00h = 0d
    endcase
end

endmodule
```

Real_Time_Digital_clock_TB.v->Test Bench

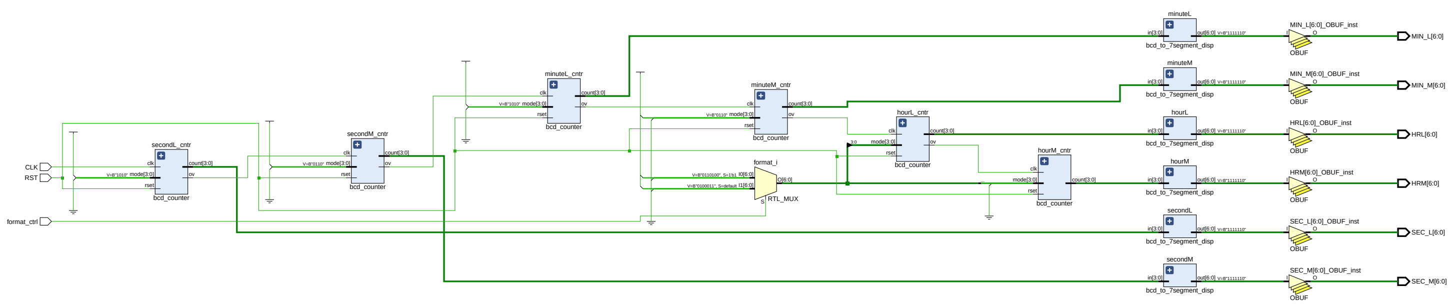
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21.08.2020 10:28:04
// Design Name:
// Module Name: Real_Time_Digital_Clock_TB
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Real_Time_Digital_Clock_TB();

wire [6:0]HRM, HRL, MIN_M, MIN_L, SEC_M, SEC_L;
reg CLK = 0;
reg RST = 1;
reg format_ctrl = 1;

Real_Time_Digital_Clock
dut(.HRM(HRM),.HRL(HRL),.MIN_M(MIN_M),.MIN_L(MIN_L),.SEC_M(SEC_M),.SEC_L(SEC_L),.CLK(CLK)
,.RST(RST),.format_ctrl(format_ctrl));

//generate clock; here Period=2ns
//This clock is given as input to the 'seconds' counter
//Therefore 2ns = 1 second
always@(CLK)
#1 CLK <= ~CLK;

initial
begin
#3 RST <= 0; //Release reset after 1.5 seconds
#86402;
format_ctrl = 0;
#43202;
$stop;
end
endmodule
```

`timescale 1ns / 1ps   **bcd_counter_TB.v --> Test Bench**

```
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 21.08.2020 10:28:04
// Design Name: 
// Module Name: bcd_counter_TB
// Project Name: 
// Target Devices: 
// Tool Versions: 
// Description: 
// 
// Dependencies: 
// 
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
// 
//////////////////////////////////////////////////////////////////////////////////


module bcd_counter_TB();

wire [3:0]count;
reg clk = 0;
reg rset = 1;
reg [3:0]mode;

bcd_counter dut( .count(count), .ov(ov), .clk(clk), .mode(mode), .rset(rset));

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

initial
begin
#12 rset <= 0;
$display("\nmod 2 counter: 0->1->0..");
mode <= 'd2; #(2*(2*10));

$display("\nmod 3 counter: 0->1->2->0");
mode <= 'd3; #(3*(2*10));

$display("\nmod 4 counter: 0->1->2->3->0");
mode <= 'd4; #(4*(2*10));

$display("\nmod 6 counter: 0->1->2->3->4->5->0");
mode <= 'd6; #(6*(2*10));

$display("\nmod 10(bcd) counter: 0->1->2->3->4->5->6->7->8->9->0");
mode <= 'd10; #(10*(2*10));
$stop;
end

initial
$monitor("clk_cycle=%d count=%d ov=%b rset=%b", ($time/10), count, ov, rset);

endmodule
```

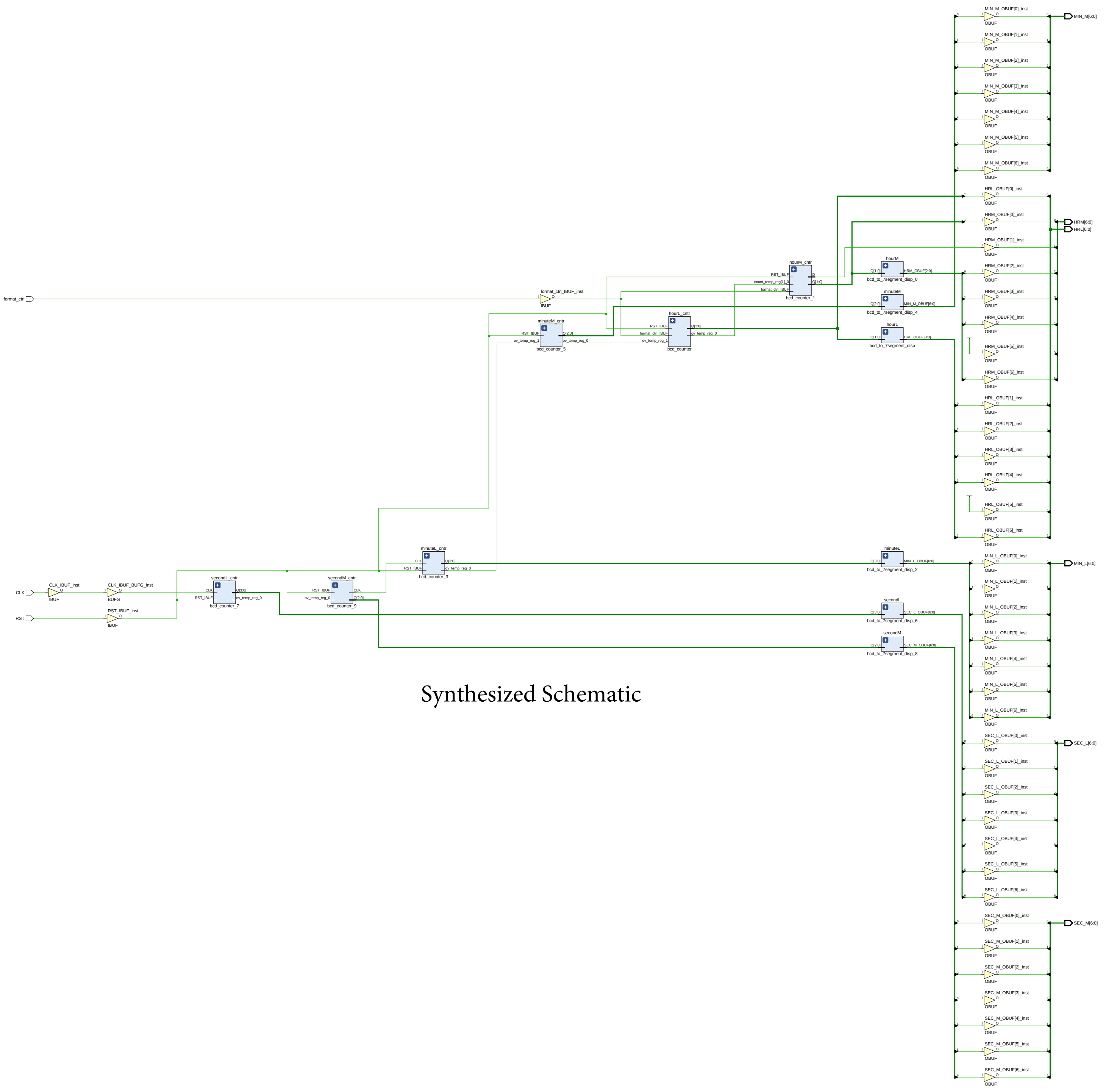
bcd_to_7segment_disp_TB.v-->Test Bench

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 21.08.2020 10:28:04
// Design Name:
// Module Name: bcd_to_7segment_disp_TB
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module bcd_to_7segment_disp_TB();

wire [6:0]y;
reg [3:0]in='d0;

bcd_to_7segment_disp dut(.out(out),.in(in));

initial
begin
    repeat(16)
    begin
    in = in + 'b1 ;
    $monitor("BCD_Disp=%b, ABCD=%b",y,in);
    #5;
    end
end

endmodule
```



RTL Schematic

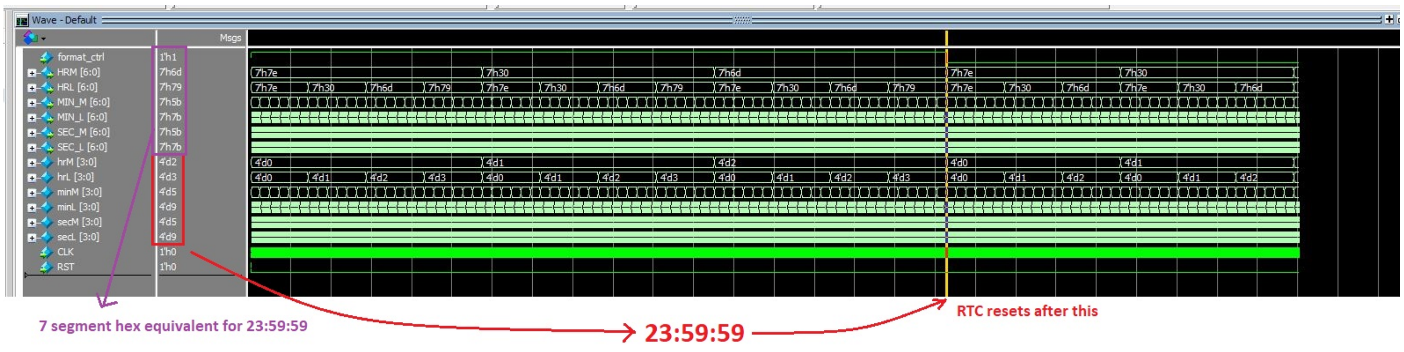
Zoom-in to view clearly



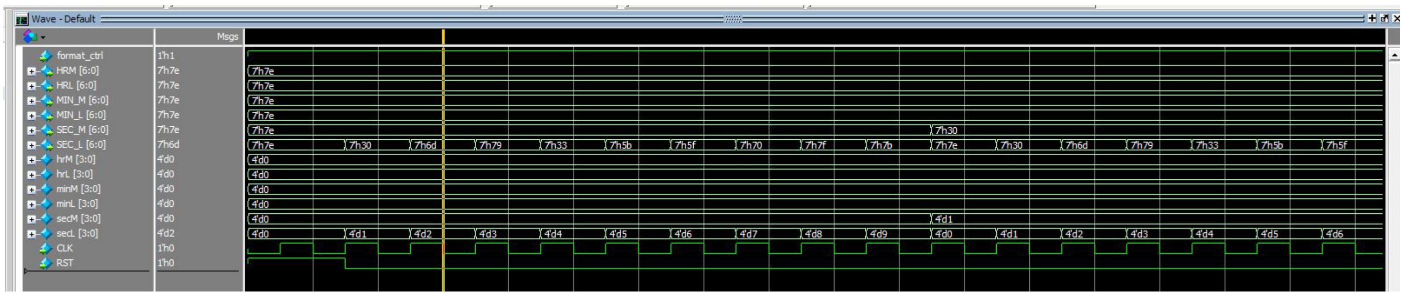
Synthesized Schematic

Zoom-in to view clearly

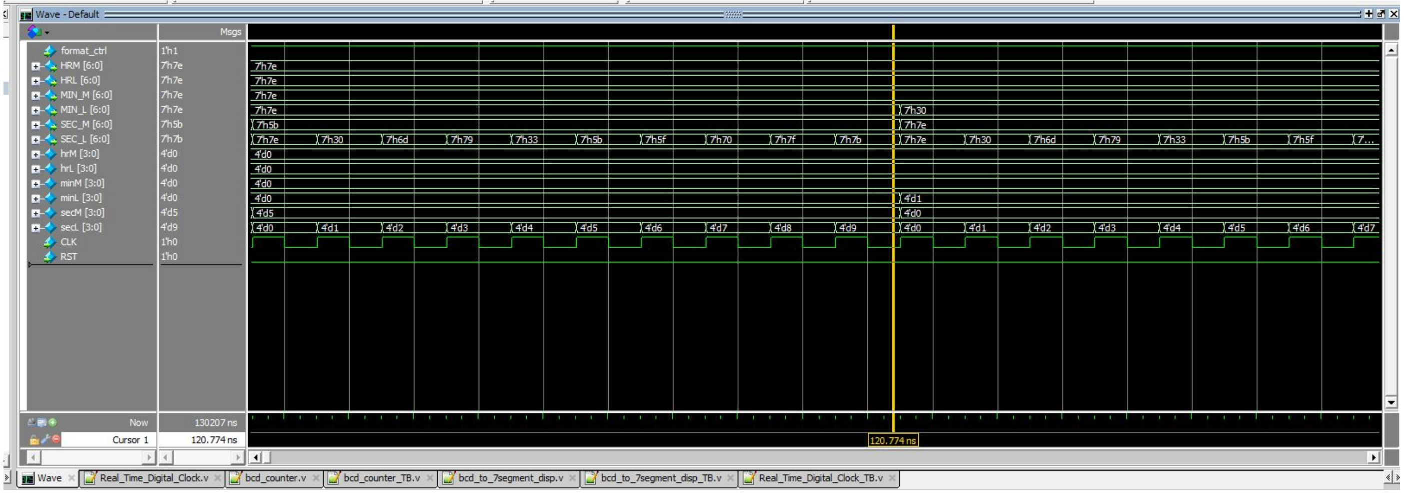
Waveform & Simulation Results



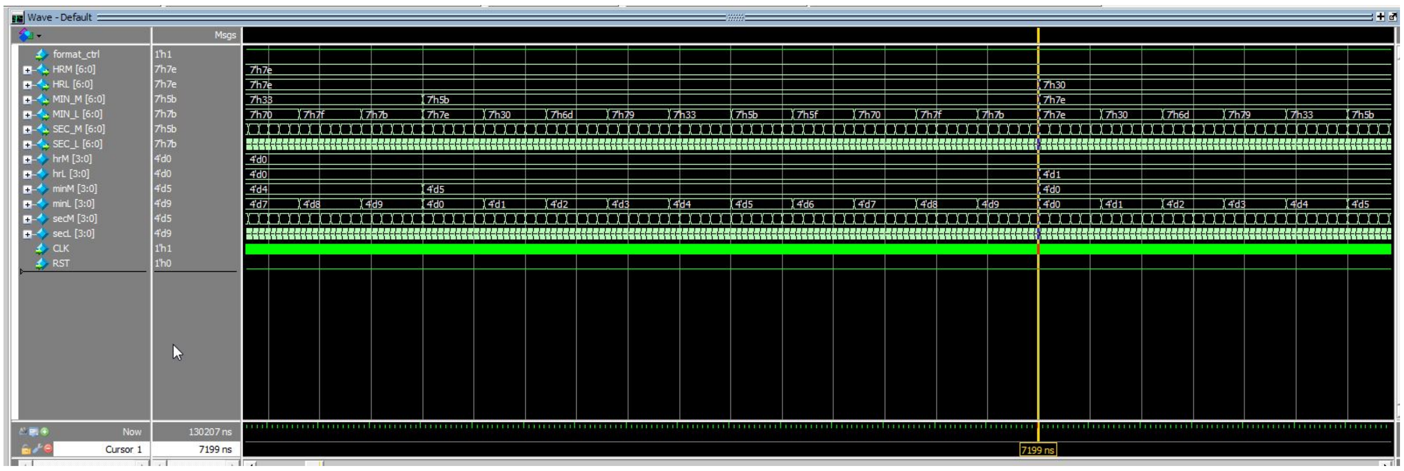
Complete_Waveform



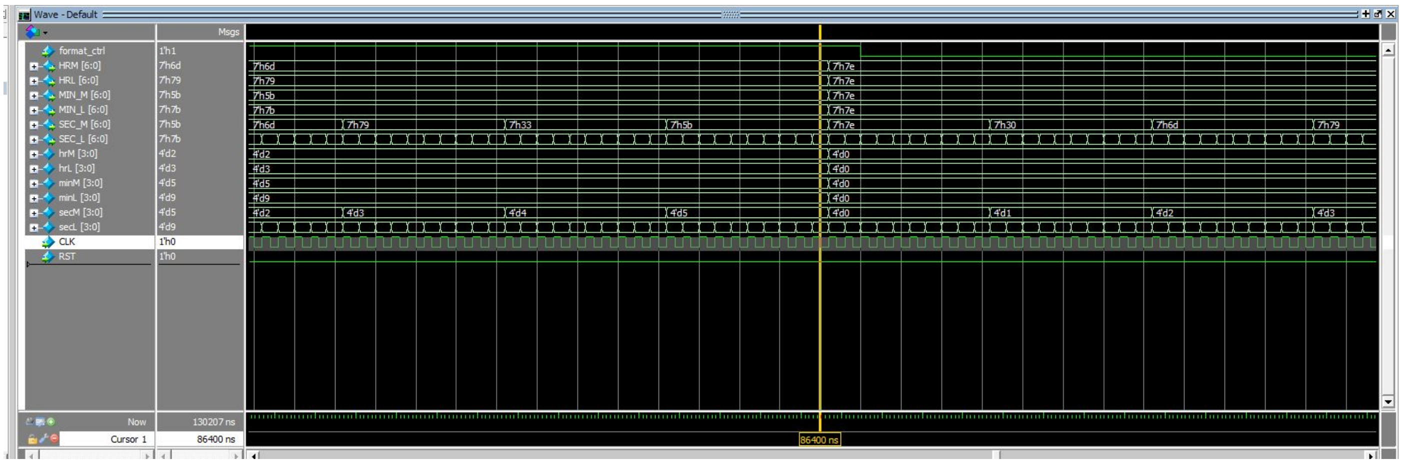
Waveform at the 'beginning of time'



59 seconds

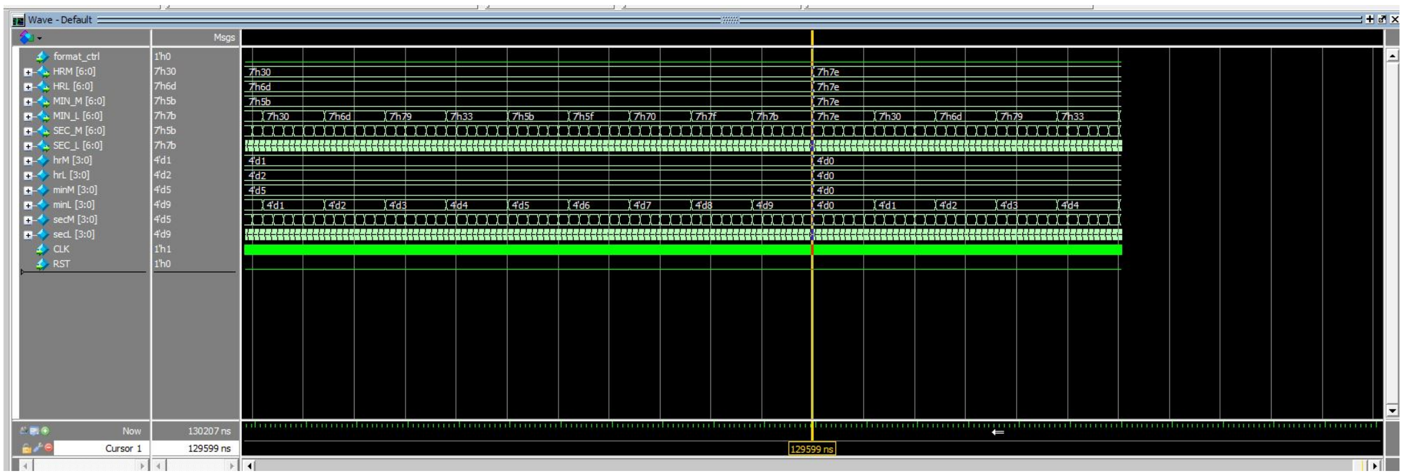


59 minutes 59 seconds



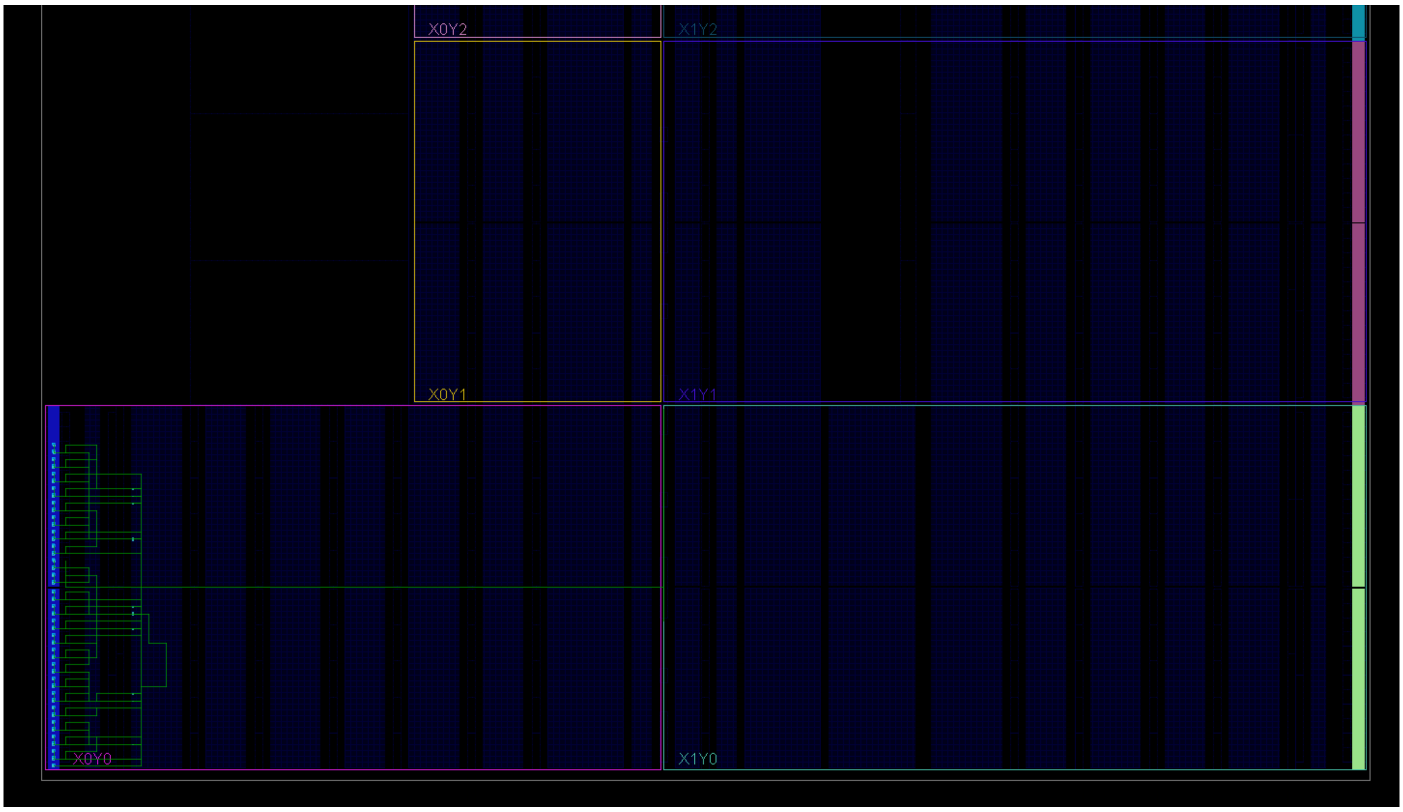
```
Wave - Default
                     Msgs
format_ctrl          1'h1
HRM [6:0]            7'h6d
HRL [6:0]            7'h79
MIN_M [6:0]          7'h5b
MIN_L [6:0]          7'h7b
SEC_M [6:0]          7'h5b
SEC_L [6:0]          7'h7b
hrM [3:0]            4'd2
hrL [3:0]            4'd3
minM [3:0]           4'd5
minL [3:0]           4'd9
secM [3:0]           4'd5
secL [3:0]           4'd9
CLK                  1'h0
RST                  1'h0
Now                  130207 ns
Cursor 1             86400 ns
```

23 Hours 59 Minutes 59 Seconds

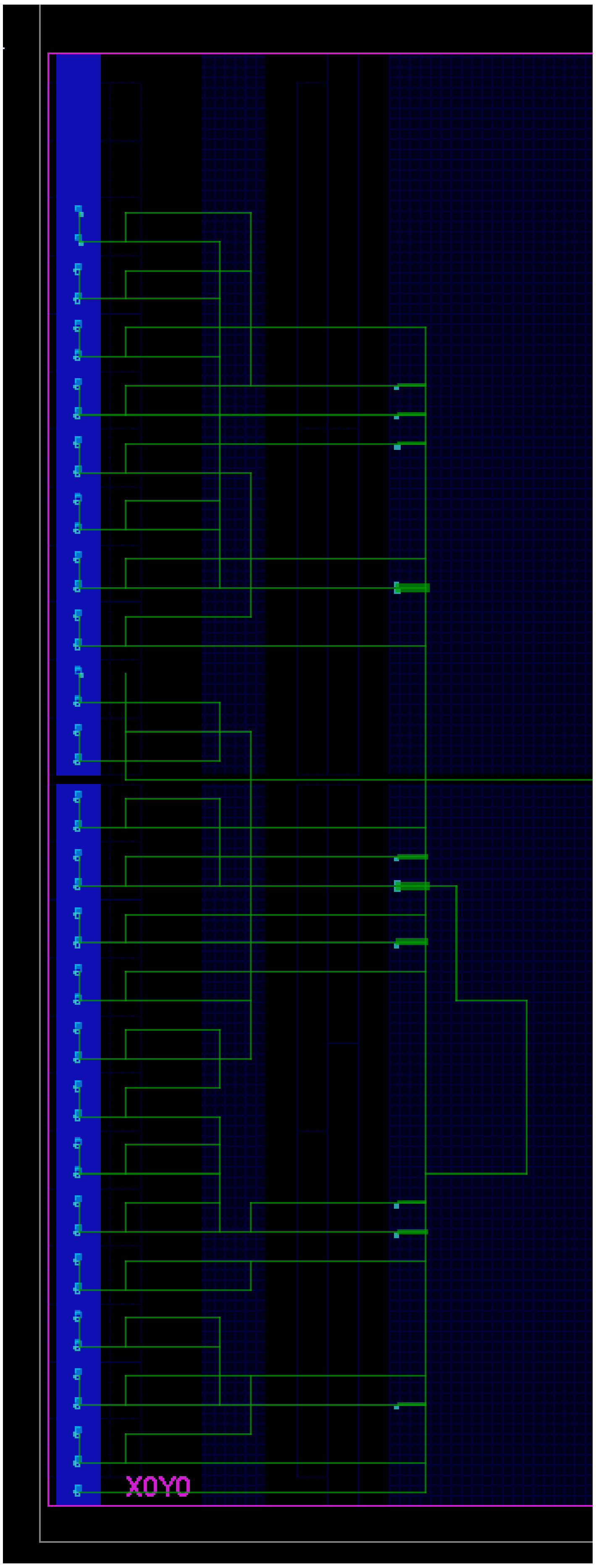


```
Wave - Default
                     Msgs
format_ctrl          1'h0
HRM [6:0]            7'h30
HRL [6:0]            7'h6d
MIN_M [6:0]          7'h5b
MIN_L [6:0]          7'h7b
SEC_M [6:0]          7'h5b
SEC_L [6:0]          7'h7b
hrM [3:0]            4'd1
hrL [3:0]            4'd2
minM [3:0]           4'd5
minL [3:0]           4'd9
secM [3:0]           4'd5
secL [3:0]           4'd9
CLK                  1'h1
RST                  1'h0
Now                  130207 ns
Cursor 1             129599 ns
```

12 Hours 59 Minutes 59 Seconds (12-Hour format)



ZED-Board (Zynq-7000) Implementation



ZED-Board (Zynq-7000) Implementation (zoomed-in view)