Visvesvaraya National Institute of Technology (VNIT), Nagpur

Digital Hardware Design (ECL313)

Project Report

Submitted by :
Domatoti Vinay Kumar (BT18ECE082)
Badrivishal Paurana (BT18ECE098)
Rishika Bhagwatkar (BT18ECE149)
Semester VI

Group No 5

Submitted to :
Dr. Anamika Singh and Dr. Neha Nawandar
(Course Instructors)
Department of Electronics and Communication Engineering,
VNIT Nagpur

Contents

Digital Watch

1.1 Theory: In this project, we have tried to simulate a digital watch on Quartus and Modelsim whose functioning is quite similar to the one shown below.

Figure 1: Watch

The watch consists of the following modules:

- Digital clock
- Stopwatch
- Alarm

The functions of the watch are controlled by 4 buttons:

- setn
- nxt
- inc
- dec

There is an internal signal “ModeW” for selecting the respective modules. Its default value is 0.

Digitalclock:

- For setting the time of watch following procedure is followed:
  - setn should be pressed (i.e its value should be 1) and mode should be 0.
  - Now for changing the Hour, Minute and Second, an internal signal "Changeval" is defined with the default value is 0.
    - When Changeval is 0, Hour changes
    - When Changeval is 1, Minute changes
    - When Changeval is 2, Second changes
  - For changing the value of Changeval, nxt button is pressed. For example, to set up the minutes to 30, setn should be 1, mode should be 0, changeval should be 1. Since changeval has a default value of 0, so we need to make nxt equal to 1 for once.
  - For changing the hours, minutes and seconds following procedure is followed
    - To increment values, inc should be 1 and dec should be 0
    - To decrement values, inc should be 0 and dec should be 1
- Once the time is set, the digital clock works normally as shown in the Fig 2.

Stopwatch:

- It is asynchronous. So it is reset whenever the reset is 1 and mode is 1.
- An internal signal "pause_play" is defined to start and stop the stopwatch. Its default value is 0. It gets inverted when inc is 1.
- Then stopwatch works over the period for which the pause_play signal is high i.e. when the duration between which inc is 1.
- The normal working of the stopwatch is shown in the Fig 3.

Alarm:

- The alarm time is set in the same manner as we set the time of the digital clock. However, the mode will be 2.
- Once the alarm time is set, it just checks whether the current time (digital clock output) with the alarm time.

- An internal signal "buzzer" goes high when the alarm time matches the current time.
- The normal working of the alarm is shown in the Fig 4.

Output:

- The output of the watch is displayed using eight 7 segment displays (2 for hour, minutes, seconds, 1 for mode and 1 for buzzer and setup).
- The order of display is HH MM SS mb.
  - HH for 2 digit hours.
  - MM for 2 digit minutes.
  - SS for 2 digit seconds.
  - mb for showing the mode and buzzer respectively.
- However, the output of which module to be displayed is controlled using the mode signal.
- If the mode is 0, the output of the digital clock will be displayed and if the mode is 1, the output of the stopwatch will be displayed.
- This enables us to easy switching between the display of outputs of the modules.
- If we are setting the time, then the last display will be showing an S (which is the same as the 5 for 7 segment display).
- Whenever the buzzer is high, the last display i.e b blinks at alternate cycles.
- **Note:** When the mode is 1 i.e stopwatch, minutes are shown in place of hours, seconds are shown in place of minutes and milliseconds are shown in place of seconds on the display until hours is 0 and when hours gets a value more than 0, the display shifts back to normal giving HH MM SS as the output. mb remains in the same position.

The flow chart depicting the working of the watch is the Fig. 5.

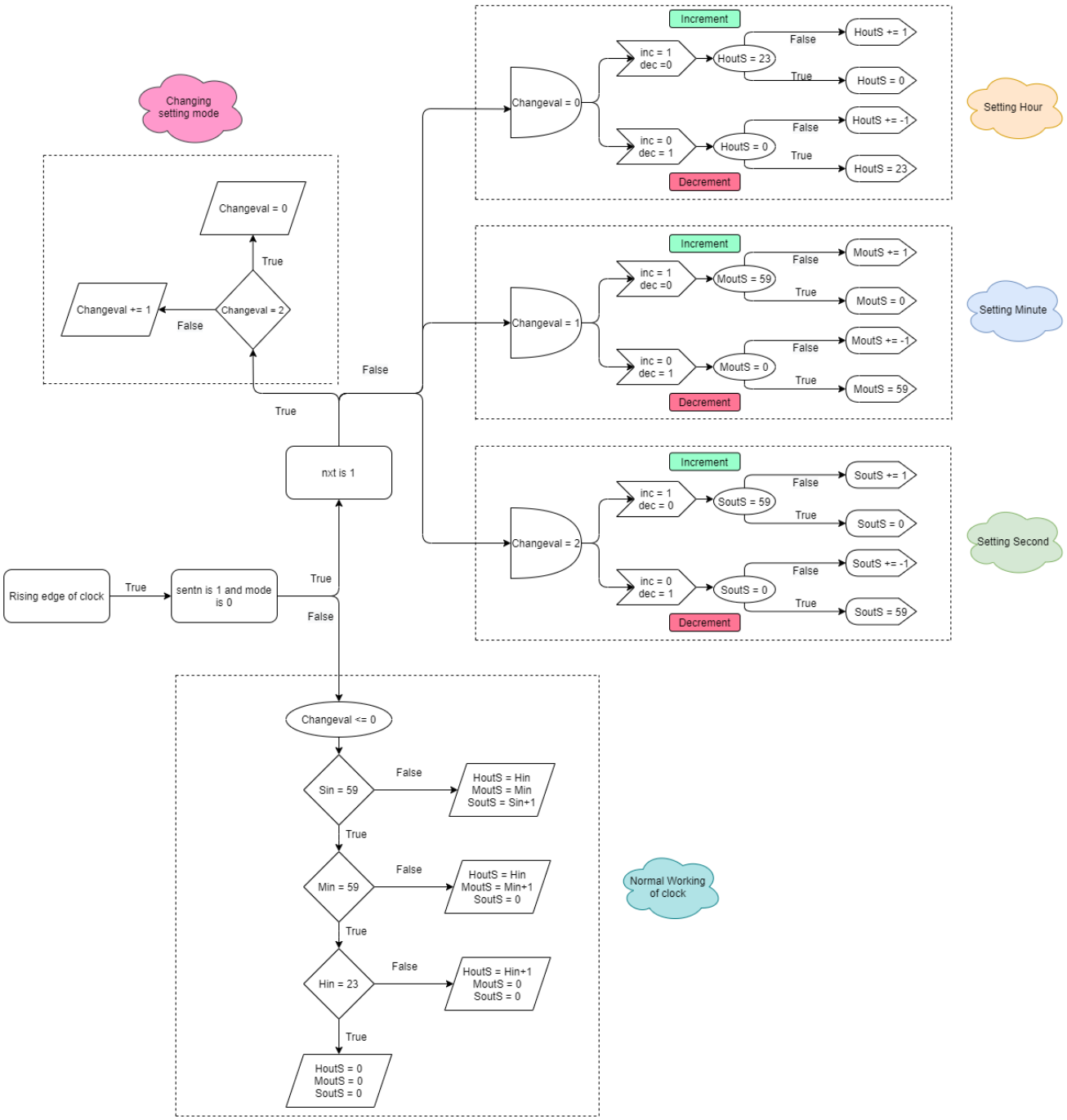

Figure 2: Flow Chart of Digital Clock

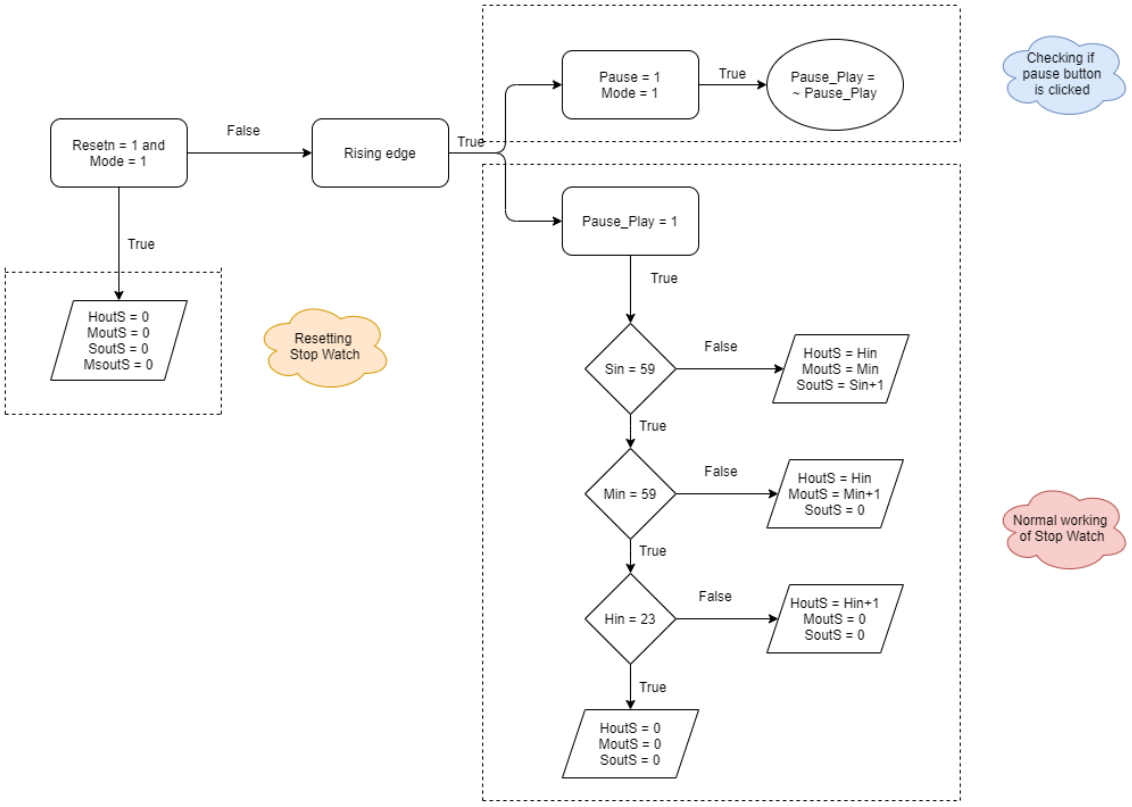

Figure 3: Flow Chart of Stopwatch

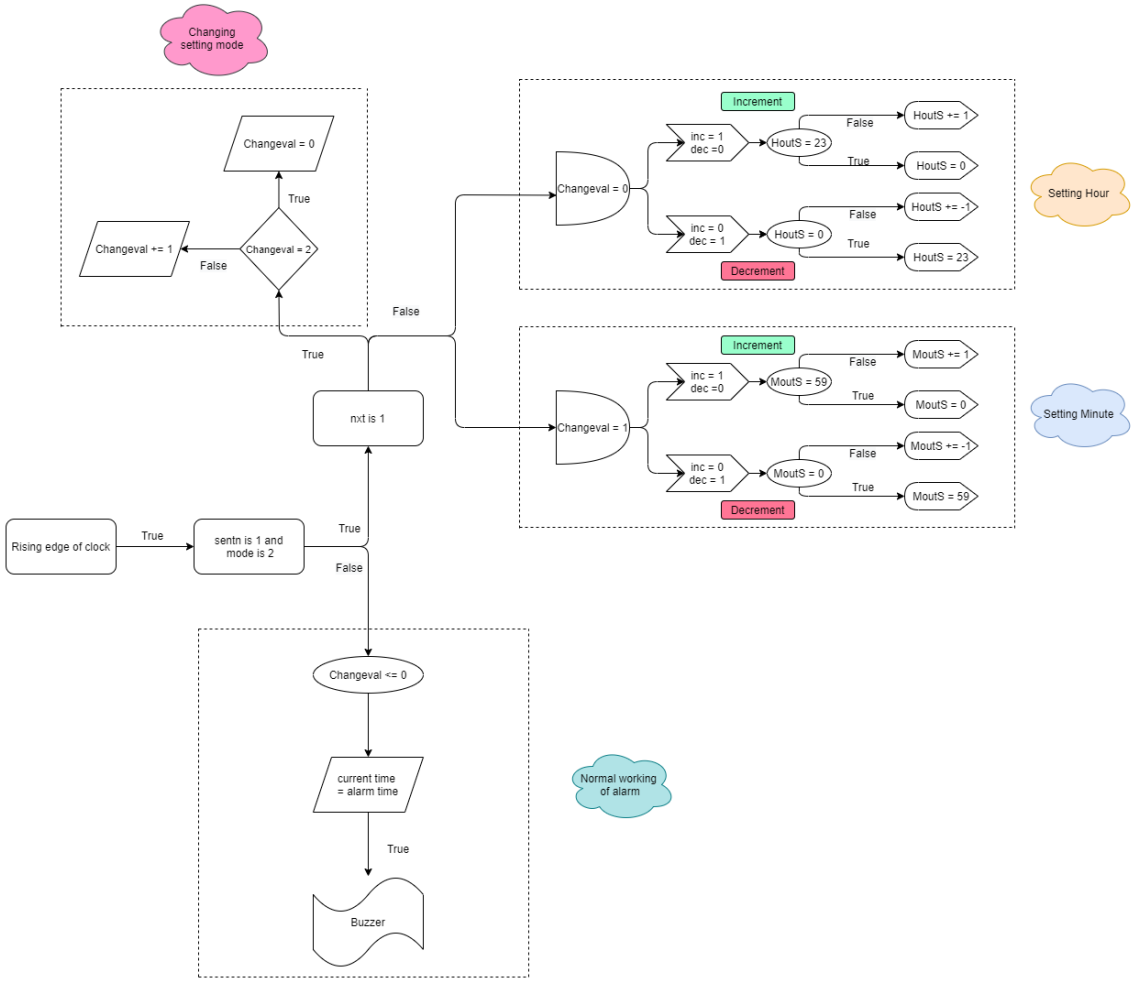

Figure 4: Flow Chart of Alarm

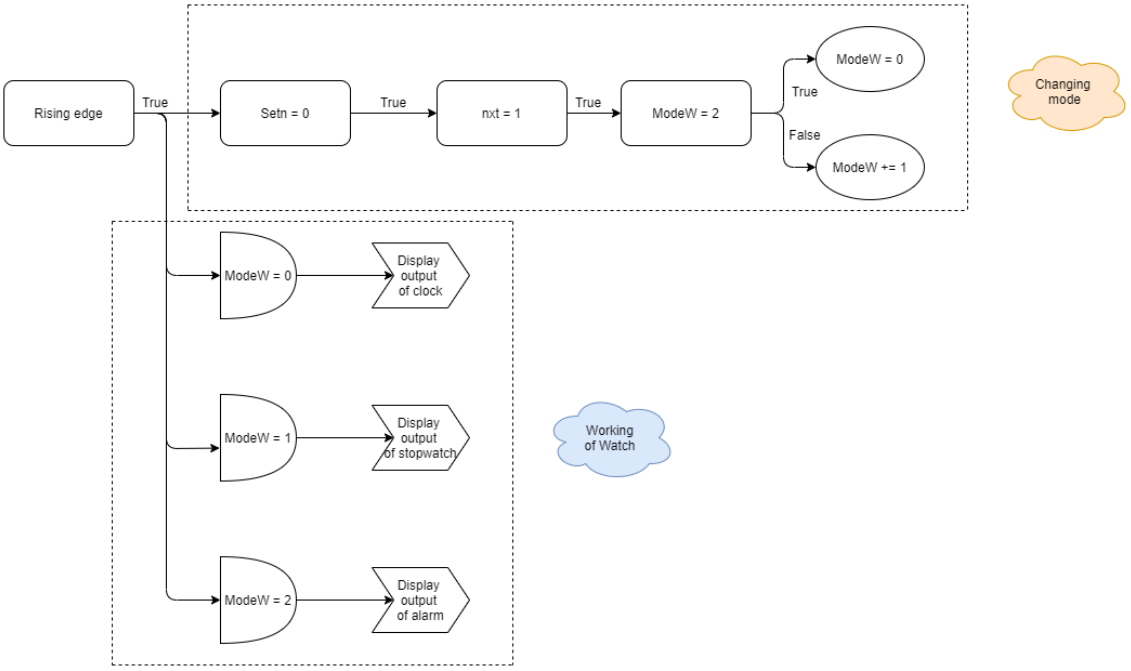

Figure 5: Flow Chart of Watch

1.2 Codes: For the coding part, the different modes were divided into different Entities.

Digital Clock

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity DigClock is
port ( clock, setn, nxt, inc, dec: in std_logic;
        mode: in integer range 0 to 2;
        Hin: inout integer range 0 to 23;
        Min: inout integer range 0 to 59;
        Sin: inout integer range 0 to 59;

        Hout: out integer range 0 to 23;
        Mout: out integer range 0 to 59;
        Sout: out integer range 0 to 59);
```

```
end DigClock;

architecture bhv of DigClock is
signal HoutS: integer range 0 to 23;
signal MoutS: integer range 0 to 59;
signal SoutS: integer range 0 to 59;
signal changeval: integer range 0 to 2;
begin
process (setn, nxt, inc, dec, clock, mode, Hin, Min, Sin)
begin
    if (clock'event and clock='1') then
        if setn = '1' and mode = 0 then
            if nxt = '1' then
                if changeval = 2 then
                    changeval <= 0;
                else
                    changeval <= changeval + 1;
                end if;
            else
                if changeval = 0 then
                    if inc = '1' and dec = '0' then
                        if HoutS = 23 then
                            HoutS <= 0;
                        else
                            HoutS <= HoutS + 1;
                        end if;
                    elsif inc = '0' and dec = '1' then
                        if HoutS = 0 then
                            HoutS <= 23;
                        else
                            HoutS <= HoutS - 1;
                        end if;
                    end if;

                elsif changeval = 1 then
                    if inc = '1' and dec = '0' then
                        if MoutS = 59 then
                            MoutS <= 0;
                        else
                            MoutS <= MoutS + 1;
```

```
                end if;
            elsif inc = '0' and dec = '1' then
                if MoutS = 0 then
                    MoutS <= 59;
                else
                    MoutS <= MoutS - 1;
                end if;
            end if;

        elsif changeval = 2 then
            if inc = '1' and dec = '0' then
                if SoutS = 59 then
                    SoutS <= 0;
                else
                    SoutS <= SoutS + 1;
                end if;
            elsif inc = '0' and dec = '1' then
                if SoutS = 0 then
                    SoutS <= 59;
                else
                    SoutS <= SoutS - 1;
                end if;
            end if;
        end if;
    end if;
else
    changeval <= 0;
    if Sin = 59 then
        if Min = 59 then
            if Hin = 23 then
                HoutS <= 0;
                MoutS <= 0;
                SoutS <= 0;
            else
                HoutS <= Hin + 1;
                MoutS <= 0;
                SoutS <= 0;
            end if;
        else
            HoutS <= Hin;
```

```
                    MoutS <= Min + 1;
                    SoutS <= 0;
                end if;
            else
                HoutS <= Hin;
                MoutS <= Min;
                SoutS <= Sin + 1;
            end if;
        end if;
    end if;
end process;
Hout <= HoutS;
Mout <= MoutS;
Sout <= SoutS;
Hin <= HoutS;
Min <= MoutS;
Sin <= SoutS;
end bhv;
```

StopWatch

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Stopwatch is
    port (clock, resetn, pause: in std_logic;
            mode: in integer range 0 to 2;
            Hin: inout integer range 0 to 23;
            Min: inout integer range 0 to 59;
            Sin: inout integer range 0 to 59;
            Msin: inout integer range 0 to 99;

            HoutA: out integer range 0 to 23;
            Mout: out integer range 0 to 59;
            Sout: out integer range 0 to 59;
            Msout: out integer range 0 to 99);
    end Stopwatch;

architecture arch of Stopwatch is
signal HoutS: integer range 0 to 23;
signal MoutS: integer range 0 to 59;
signal SoutS: integer range 0 to 59;
signal MsoutS: integer range 0 to 99;
signal pause_play: std_logic := '0';
begin
    process (resetn, pause, clock, mode, Hin, Min, Sin, Msin)
    begin

        if resetn = '1' and mode = 1 then
            pause_play <= '0';
            HoutS <= 0;
            MoutS <= 0;
            SoutS <= 0;
            MsoutS <= 0;
        else
            if (clock'event and clock='1') then
                if pause = '1' and mode = 1 then
                    pause_play <= not pause_play;
```

```
                end if;
                if pause_play = '1' then
                    if Msin = 99 then
                        if Sin = 59 then
                            if Min = 59 then
                                if Hin = 23 then
                                    HoutS <= 0;
                                    MoutS <= 0;
                                    SoutS <= 0;
                                    MsoutS <= 0;
                                else
                                    HoutS <= Hin + 1;
                                    MoutS <= 0;
                                    SoutS <= 0;
                                    MsoutS <= 0;
                                end if;
                            else
                                HoutS <= Hin;
                                MoutS <= Min + 1;
                                SoutS <= 0;
                                MsoutS <= 0;
                            end if;
                        else
                            HoutS <= Hin;
                            MoutS <= Min;
                            SoutS <= Sin + 1;
                            MsoutS <= 0;
                        end if;
                    else
                        HoutS <= Hin;
                        MoutS <= Min;
                        SoutS <= Sin;
                        MsoutS <= Msin + 1;
                    end if;
                end if;
            end if;
        end if;
    end process;
    HoutA <= HoutS;
    Mout <= MoutS;
```

```
    Sout <= SoutS;
    Msout <= MsoutS;
    Hin <= HoutS;
    Min <= MoutS;
    Sin <= SoutS;
    Msin <= MsoutS;
end arch;
```

Alarm

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Alarm is
port (clock, setn, nxt, inc, dec: in std_logic;
        mode: in integer range 0 to 2;
        Hin: in integer range 0 to 23;
        Min: in integer range 0 to 59;
        Sin: in integer range 0 to 59;

        HAl: out integer range 0 to 23;
        MAl: out integer range 0 to 59;
        buzzer: out std_logic);
end Alarm;

architecture arch of Alarm is
signal HAlS: integer range 0 to 23;
signal MAlS: integer range 0 to 59;
signal changeval: integer range 0 to 1;
signal buzzerS: std_logic := '0';
begin
process(setn, nxt, inc, dec, clock, mode, Hin, Min, Sin)
begin
    if (clock'event and clock='1') then
        if setn = '1' and mode = 2 then
            if nxt = '1' then
                if changeval = 1 then
                    changeval <= 0;
                else
                    changeval <= changeval + 1;
                end if;
            else
                if changeval = 0 then
                    if inc = '1' and dec = '0' then
                        if HAlS = 23 then
                            HAlS <= 0;
                        else
```

```
                        HAlS <= HAlS + 1;
                    end if;
                elsif inc = '0' and dec = '1' then
                    if HAlS = 0 then
                        HAlS <= 23;
                    else
                        HAlS <= HAlS - 1;
                    end if;
                end if;

            elsif changeval = 1 then
                if inc = '1' and dec = '0' then
                    if MAlS = 59 then
                        MAlS <= 0;
                    else
                        MAlS <= MAlS + 1;
                    end if;
                elsif inc = '0' and dec = '1' then
                    if MAlS = 0 then
                        MAlS <= 59;
                    else
                        MAlS <= MAlS - 1;
                    end if;
                end if;
            end if;
        else
            changeval <= 0;
            if (Hin = HAlS and Min = MAlS) then
                buzzerS <= '1';
            else
                buzzerS <= '0';
            end if;
        end if;
    end if;
end process;
HAl <= HAlS;
MAl <= MAlS;
buzzer <= buzzerS;
end arch;
```

Watch

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity Watch is
port (clock, ClockSt, setn, nxt, inc, dec: in std_logic;
    Hdisp1, Hdisp2, Mdisp1, Mdisp2, Sdisp1, Sdisp2, Modedisp1, Modedisp2: out std_l
    buzzer: out std_logic);
end Watch;

architecture arch of Watch is

    component DigClock
    port ( clock, setn, nxt, inc, dec: in std_logic;
        mode: in integer range 0 to 2;
        Hin: inout integer range 0 to 23;
        Min: inout integer range 0 to 59;
        Sin: inout integer range 0 to 59;

        Hout: out integer range 0 to 23;
        Mout: out integer range 0 to 59;
        Sout: out integer range 0 to 59);
    end component;

    component Stopwatch
    port (clock, resetn, pause: in std_logic;
        mode: in integer range 0 to 2;
        Hin: inout integer range 0 to 23;
        Min: inout integer range 0 to 59;
        Sin: inout integer range 0 to 59;
        Msin: inout integer range 0 to 99;

        HoutA: out integer range 0 to 23;
        Mout: out integer range 0 to 59;
        Sout: out integer range 0 to 59;
        Msout: out integer range 0 to 99);
```

```
end component;

component Alarm
port (clock, setn, nxt, inc, dec: in std_logic;
    mode: in integer range 0 to 2;
    Hin: in integer range 0 to 23;
    Min: in integer range 0 to 59;
    Sin: in integer range 0 to 59;

    HAl: out integer range 0 to 23;
    MAl: out integer range 0 to 59;
    buzzer: out std_logic);
end component;

-- Clock Variables
signal HinClk: integer range 0 to 23;
signal MinClk: integer range 0 to 59;
signal SinClk: integer range 0 to 59;

signal HoutClk: integer range 0 to 23;
signal MoutClk: integer range 0 to 59;
signal SoutClk: integer range 0 to 59;


-- StopWatch Variables
signal HinSt: integer range 0 to 23;
signal MinSt: integer range 0 to 59;
signal SinSt: integer range 0 to 59;
signal MsinSt: integer range 0 to 99;

signal HoutSt: integer range 0 to 23;
signal MoutSt: integer range 0 to 59;
signal SoutSt: integer range 0 to 59;
signal MsoutSt: integer range 0 to 99;

-- Alarm Variables
signal HinAl: integer range 0 to 23;
signal MinAl: integer range 0 to 59;
signal SinAl: integer range 0 to 59;
```

```
signal HAl: integer range 0 to 23;
signal MAl: integer range 0 to 59;
signal buzzerS: std_logic := '0';

Signal modeW: integer range 0 to 2 := 0;

type bit7 is array (0 to 11) of std_logic_vector(6 downto 0);
Constant seg7disp : bit7 := ("1111110", "0110000", "1101101",
"1111001", "0110011", "1011011", "1011111", "1110000", "1111111",
"1110011", "0000000", "0011111");

Signal HdispS1, HdispS2, MdispS1, MdispS2, SdispS1, SdispS2, ModedispS1,
ModedispS2: std_logic_vector(6 downto 0);

begin
u0: DigClock port map(clock => clock, setn => setn, nxt => nxt,
        inc => inc, dec => dec, mode => modeW, Hin => HinClk,
        Min => MinClk,Sin => SinClk, Hout => HoutClk,
        Mout => MoutClk, Sout => SoutClk);

u1: Stopwatch port map(clock => ClockSt, resetn => setn,
        pause => inc, mode => modeW, Hin => HinSt, Min => MinSt,
        Sin => SinSt, Msin => MsinSt, HoutA => HoutSt,
        Mout =>MoutSt, Sout => SoutSt, Msout => MsoutSt);

u2: Alarm port map(clock => clock, setn => setn, nxt => nxt,
        inc => inc, dec => dec, mode => modeW, Hin => HinAl,
        Min => MinAl, Sin => SinAl, HAl => HAl,
        MAl => MAl, buzzer => buzzerS);

process(clock, setn, nxt, inc, dec)
begin
    if (clock'event and clock='1') then
        if setn = '0' then
            if nxt = '1' then
                if modeW = 2 then
                    modeW <= 0;
                else
                    modeW <= modeW + 1;
                end if;
```

```
        end if;
    end if;


    if modeW = 0 then
        SdispS1 <= seg7disp(SinClk/10);
        SdispS2 <= seg7disp(SinClk mod 10);
        MdispS1 <= seg7disp(MinClk/10);
        MdispS2 <= seg7disp(MinClk mod 10);
        HdispS1 <= seg7disp(HinClk/10);
        HdispS2 <= seg7disp(HinClk mod 10);
        ModedispS1 <= seg7disp(0);

    elsif modeW = 1 then
        if HoutSt = 0 then
            SdispS1 <= seg7disp(MsoutSt/10);
            SdispS2 <= seg7disp(MsoutSt mod 10);
            MdispS1 <= seg7disp(SoutSt/10);
            MdispS2 <= seg7disp(SoutSt mod 10);
            HdispS1 <= seg7disp(MoutSt/10);
            HdispS2 <= seg7disp(MoutSt mod 10);
        else
            SdispS1 <= seg7disp(SoutSt/10);
            SdispS2 <= seg7disp(SoutSt mod 10);
            MdispS1 <= seg7disp(MoutSt/10);
            MdispS2 <= seg7disp(MoutSt mod 10);
            HdispS1 <= seg7disp(HoutSt/10);
            HdispS2 <= seg7disp(HoutSt mod 10);
        end if;
        ModedispS1 <= seg7disp(1);
    else
        SdispS1 <= seg7disp(10);
        SdispS2 <= seg7disp(10);
        MdispS1 <= seg7disp(MAl/10);
        MdispS2 <= seg7disp(MAl mod 10);
        HdispS1 <= seg7disp(HAl/10);
        HdispS2 <= seg7disp(HAl mod 10);
        ModedispS1 <= seg7disp(2);
    end if;
```

```
            if buzzerS = '1' and setn = '0' then
                if SoutClk mod 2 = 0 then
                    ModedispS2 <= seg7disp(11);
                else
                    ModedispS2 <= seg7disp(10);
                end if;
            else
                ModedispS2 <= seg7disp(10);
            end if;

            if setn = '1' then
                ModedispS2 <= seg7disp(5);
            end if;

        end if;
    end process;

    HinAl <= HoutClk;
    MinAl <= MoutClk;
    SinAl <= SoutClk;

    Hdisp1 <= HdispS1;
    Hdisp2 <= HdispS2;
    Mdisp1 <= MdispS1;
    Mdisp2 <= MdispS2;
    Sdisp1 <= SdispS1;
    Sdisp2 <= SdispS2;
    Modedisp1 <= ModedispS1;
    Modedisp2 <= ModedispS2;

    buzzer <= buzzerS;
end arch;
```

1.3 Testbench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Watch_tb IS
end Watch_tb;

architecture arch of Watch_tb IS
    component Watch
        port(setn :  in std_logic;
            nxt :    in std_logic;
            inc :    in std_logic;
            dec :    in std_logic;
            clock :  in std_logic;
            ClockSt :in std_logic;
            Hdisp1 :    out std_logic_vector(6 downto 0);
            Hdisp2 :    out std_logic_vector(6 downto 0);
            Mdisp1 :    out std_logic_vector(6 downto 0);
            Mdisp2 :    out std_logic_vector(6 downto 0);
            Sdisp1 :    out std_logic_vector(6 downto 0);
            Sdisp2 :    out std_logic_vector(6 downto 0);
            Modedisp1 : out std_logic_vector(6 downto 0);
            Modedisp2 : out std_logic_vector(6 downto 0);
            buzzer :    out std_logic
                );
        end component;
        signal ClockSt: std_logic;
        signal setn   : std_logic :='0';
        signal nxt    : std_logic :='0';
        signal inc    : std_logic :='0';
        signal dec    : std_logic :='0';
        signal clock  : std_logic ;
        signal Hdisp1 : std_logic_vector(6 downto 0);
        signal Hdisp2 : std_logic_vector(6 downto 0);
        signal Mdisp1 : std_logic_vector(6 downto 0);
        signal Mdisp2 : std_logic_vector(6 downto 0);
        signal Sdisp1 : std_logic_vector(6 downto 0);
        signal Sdisp2    : std_logic_vector(6 downto 0);
```

```
        signal Modedisp1 : std_logic_vector(6 downto 0);
        signal Modedisp2 : std_logic_vector(6 downto 0);
        signal buzzer    : std_logic;
        constant clk_period   : time := 200 ps;
        constant clkSt_period : time := 2 ps;
BEGIN
        uut: Watch port map(
                setn => setn,
                nxt        => nxt,
                inc        => inc,
                dec        => dec,
                clock => clock,
                ClockSt => ClockSt,
                Hdisp1 => Hdisp1,
                Hdisp2 => Hdisp2,
                Mdisp1 => Mdisp1,
                Mdisp2 => Mdisp2,
                Sdisp1 => Sdisp1,
                Sdisp2 => Sdisp2,
                Modedisp1 => Modedisp1,
                Modedisp2 => Modedisp2,
                buzzer => buzzer
                );

        clk_process :process
                begin
                clock <= '1';
                wait for clk_period/2;
                clock <= '0';
                wait for clk_period/2;
                end process;

        clkSt_process: process
                begin
                ClockSt <= '1';
                wait for clkSt_period/2;
                ClockSt <= '0';
                wait for clkSt_period/2;
                end process;
```

```
watch_process: process
   begin
        setn <= '0';
        nxt <= '0';
        inc <='0';
        dec <='0';
        wait for 600 ps;
        -- runs Time for 3secs

        setn <= '1';
        dec <= '1';
        wait for 600 ps;
        -- sets Time to 21hrs

        dec <= '0';
        inc <= '1';
        wait for 400 ps;
        -- sets Time to 23hrs

        nxt <= '1';
        wait for 200 ps;
        -- setter changed to min

        nxt <= '0';
        inc <= '0';
        dec <= '1';
        wait for 400 ps;
        -- set Time as 58min

        nxt <= '1';
        wait for 200 ps;
        -- setter changed to secs

        nxt <= '0';
        dec <= '0';
        inc <= '1';
        wait for 1000 ps;
        -- set Time for 5secs (23.58.8)

        setn <= '0';
```

```
inc <= '0';
wait for 400 ps;
-- run normal watch for 2 secs (23.58.10)

nxt <= '1';
wait for 200 ps;
-- change mode to Stopwatch

nxt <= '0';
inc <= '1';
wait for 2 ps;
-- pressed pause_play stopwatch

inc <= '0';
wait for 998 ps;
-- run stopwatch for 5secs (4 99)

nxt <= '1';
wait for 200 ps;
-- change mode to Alarm

nxt <= '0';
setn <= '1';
dec <= '1';
wait for 200 ps;
-- set Alarm at 23Hrs

nxt <= '1';
wait for 200 ps;
-- setter changed to Minutes

nxt <= '0';
wait for 200 ps;
-- set Alarm to 59 Minutes

setn <= '0';
nxt <= '1';
wait for 200 ps;
-- Mode set to Time
```

```
                nxt <= '0';
                wait for 8600 ps;
                -- Alarm Will ring

                nxt <= '1';
                wait for 200 ps;
                -- change mode to stopwatch

                nxt <= '0';
                inc <= '1';
                wait for 2 ps;
                -- paused

                inc <= '0';
                wait for 598 ps;
                -- stayed paused 3secs

                inc <= '1';
                wait for 2 ps;
                -- resumed

                inc <= '0';
                wait for 398 ps;
                -- resumed for 2 secs

                setn <= '1';
                wait for 200 ps;
                -- reset

                setn <= '0';
                nxt <= '1';
                wait for 400 ps;
                -- Back to time

                nxt <= '0';
                wait for 10000 ps;
                -- 23 to 00
                wait;
        end process;
END ;
```

1.4 RTL Schematic: By simulating the code above, we get the following RTL view

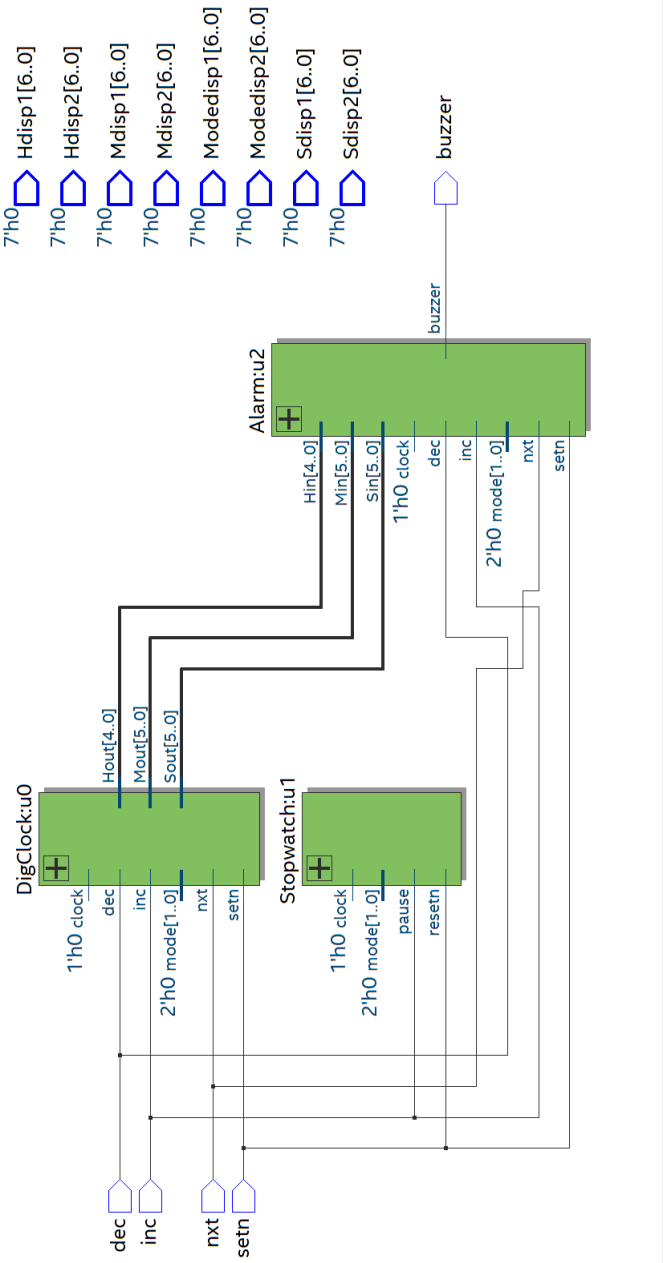

Figure 6: RTL view for Watch

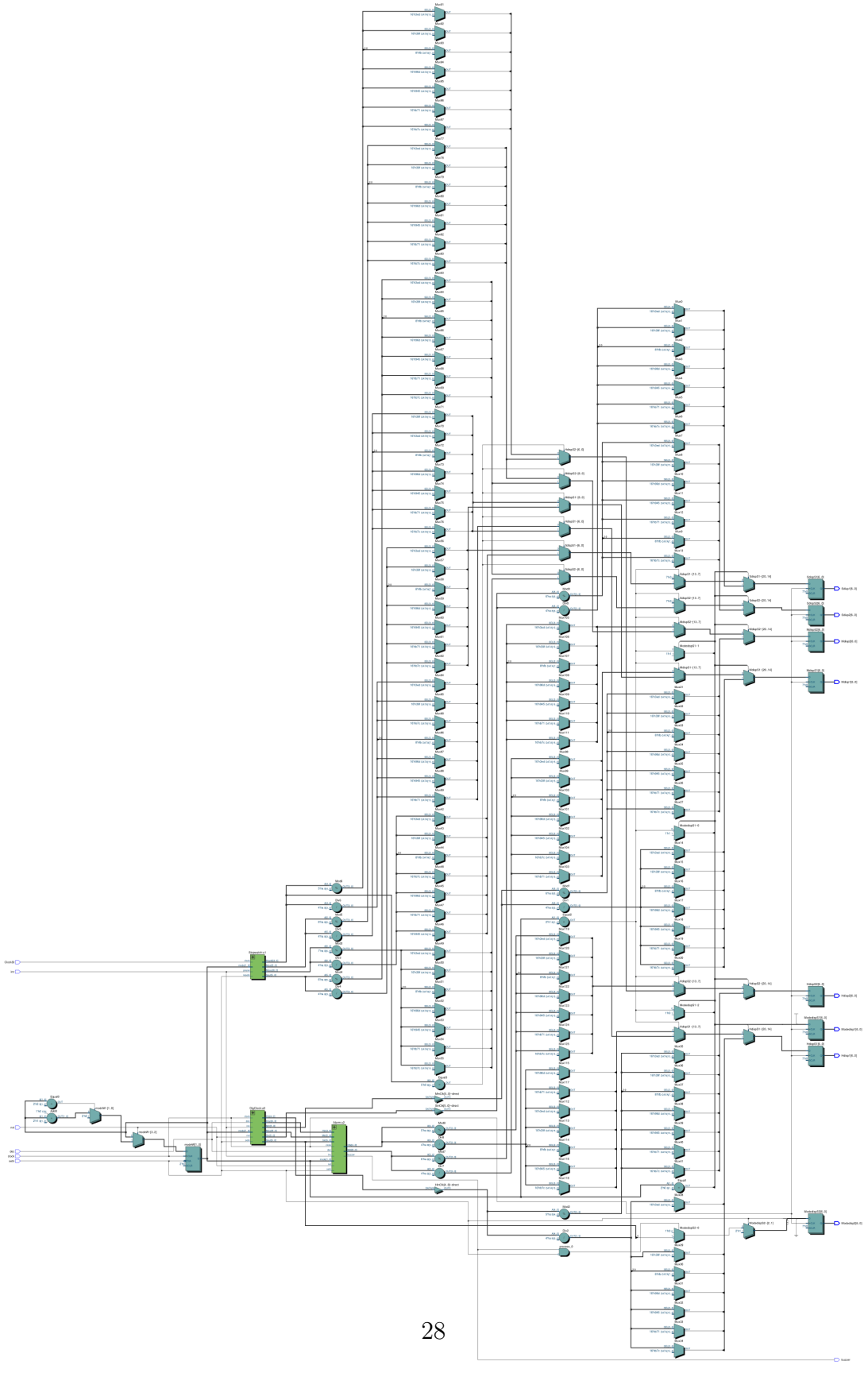

Figure 7: RTL view for Watch

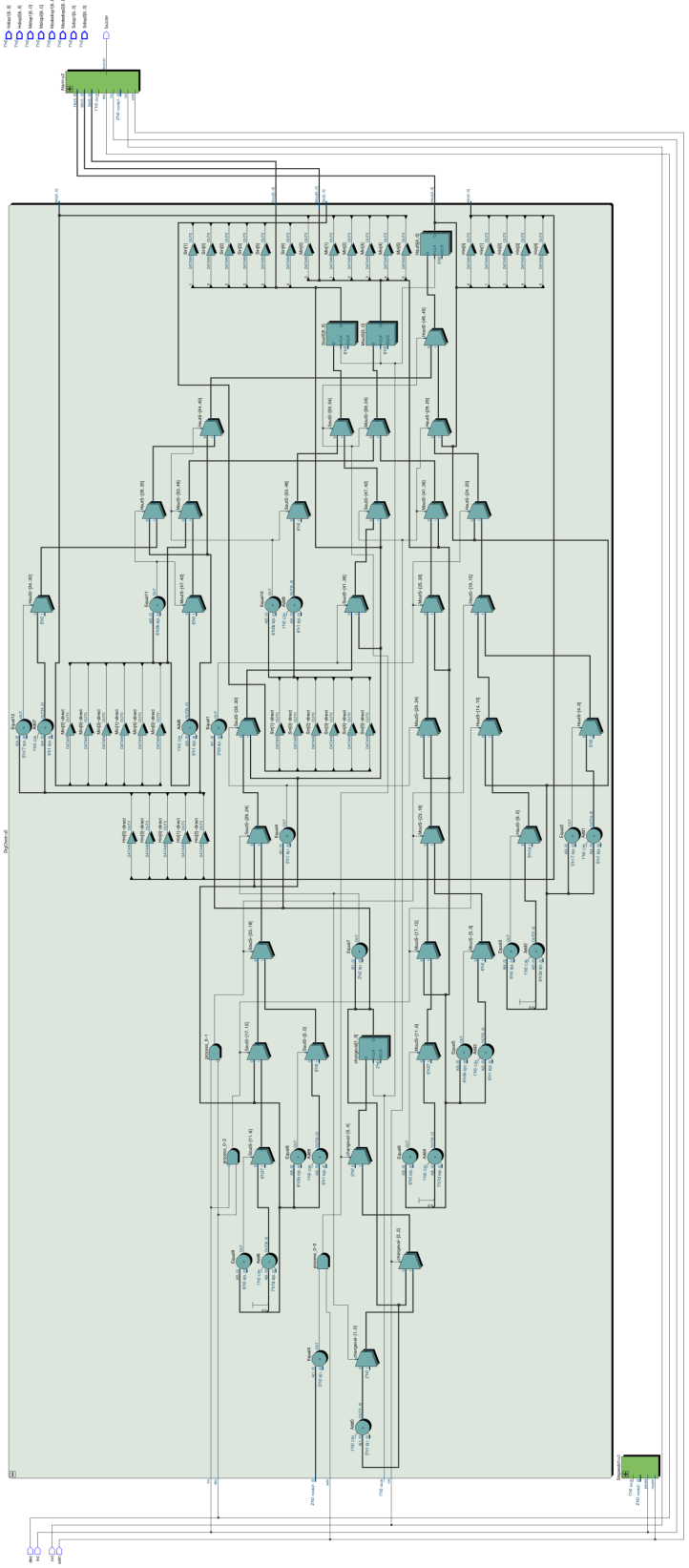

Figure 8: RTL view for Digital Clock

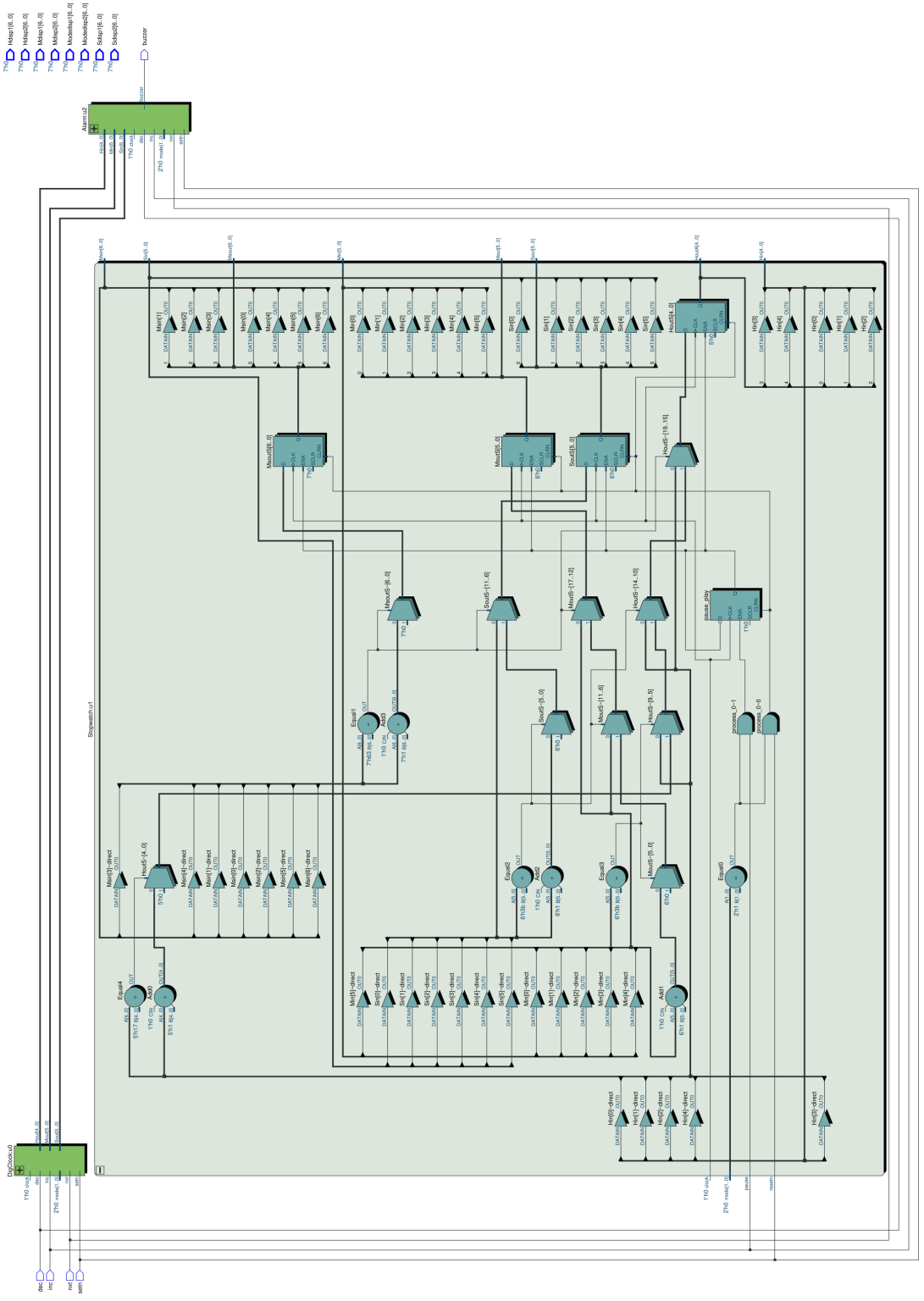

Figure 9: RTL view for StopWatch

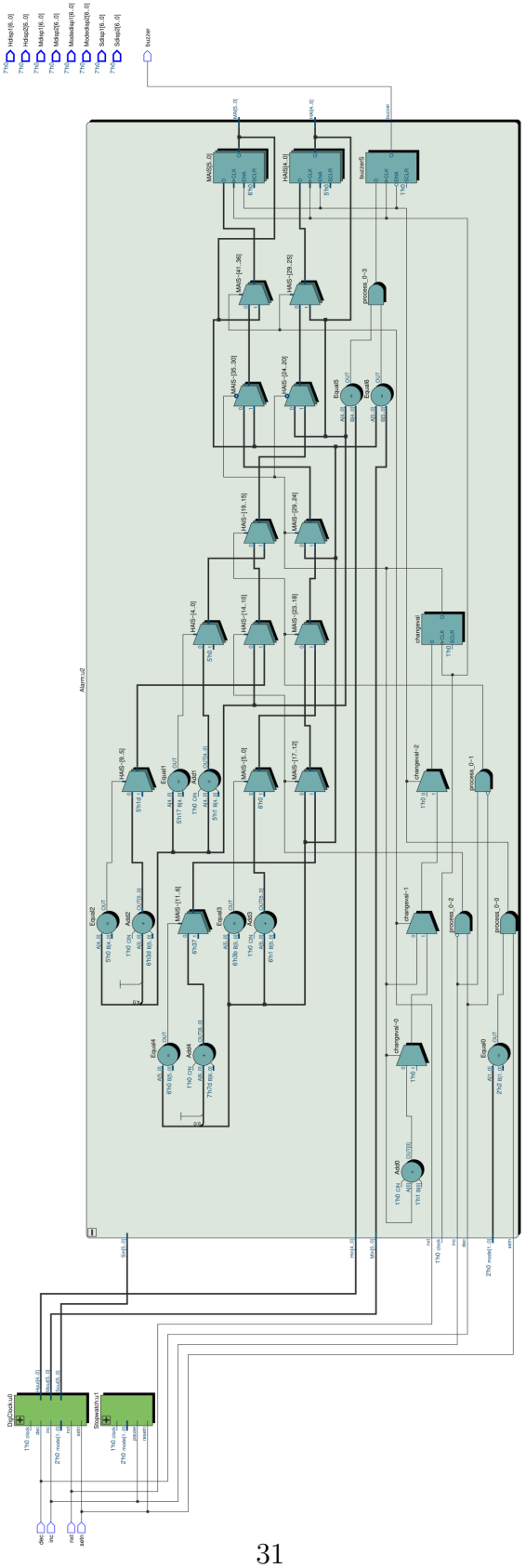

Figure 10: RTL view for Alarm

1.5 Simulation Waverform: The following are the simulation outputs

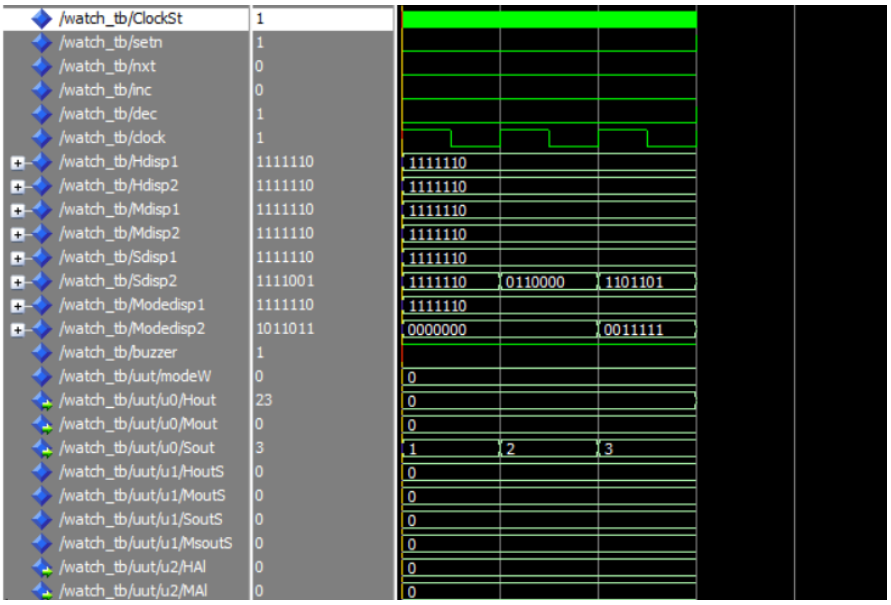

Figure 11: runtime = 600ps
Runs clock for 3secs, displayed in Sdisp1 and Sdisp2.

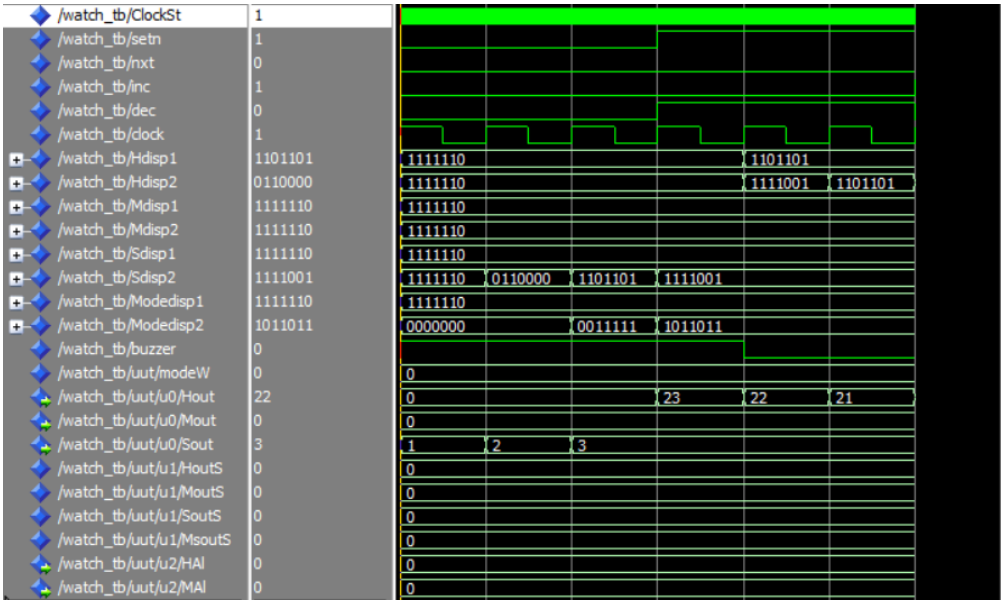

Figure 12: Runtime = 600ps
Mode set to change Hours: Hours set to 21.

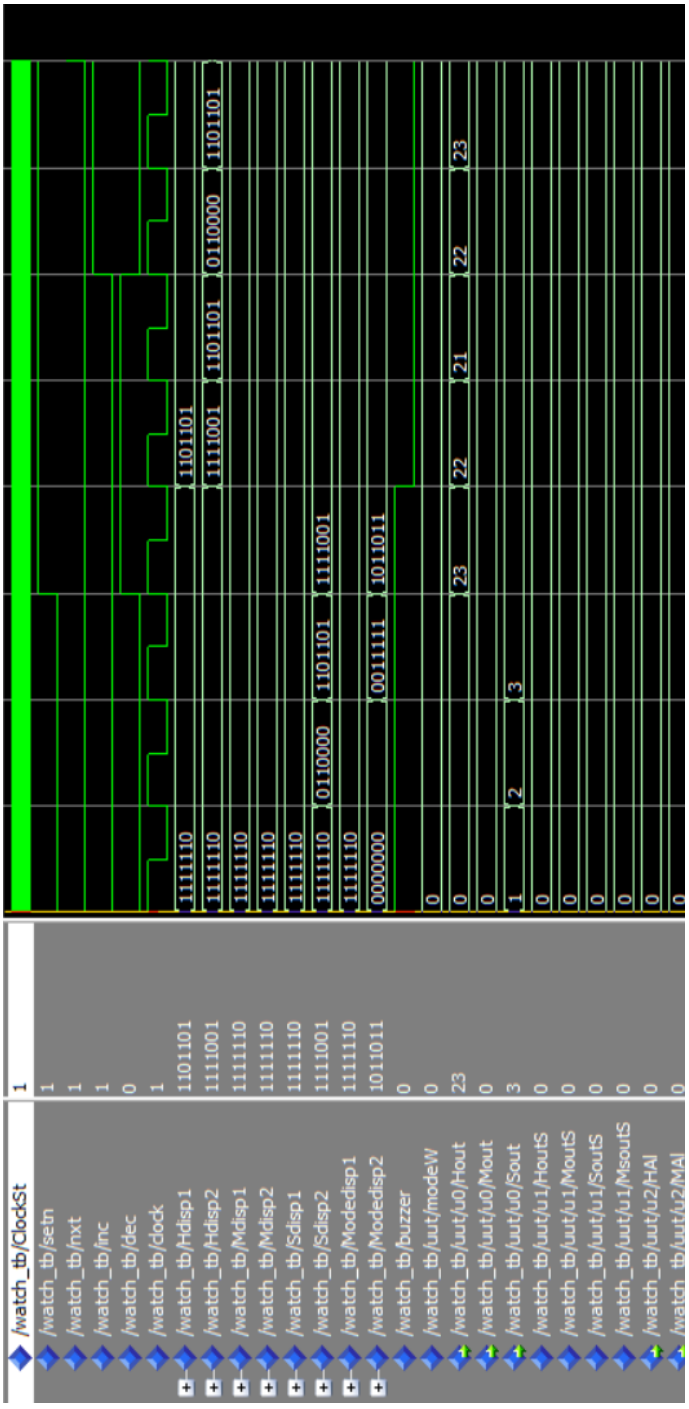

Figure 13: Runtime = 400ps
Mode set to change Hours: Hours set to 23.

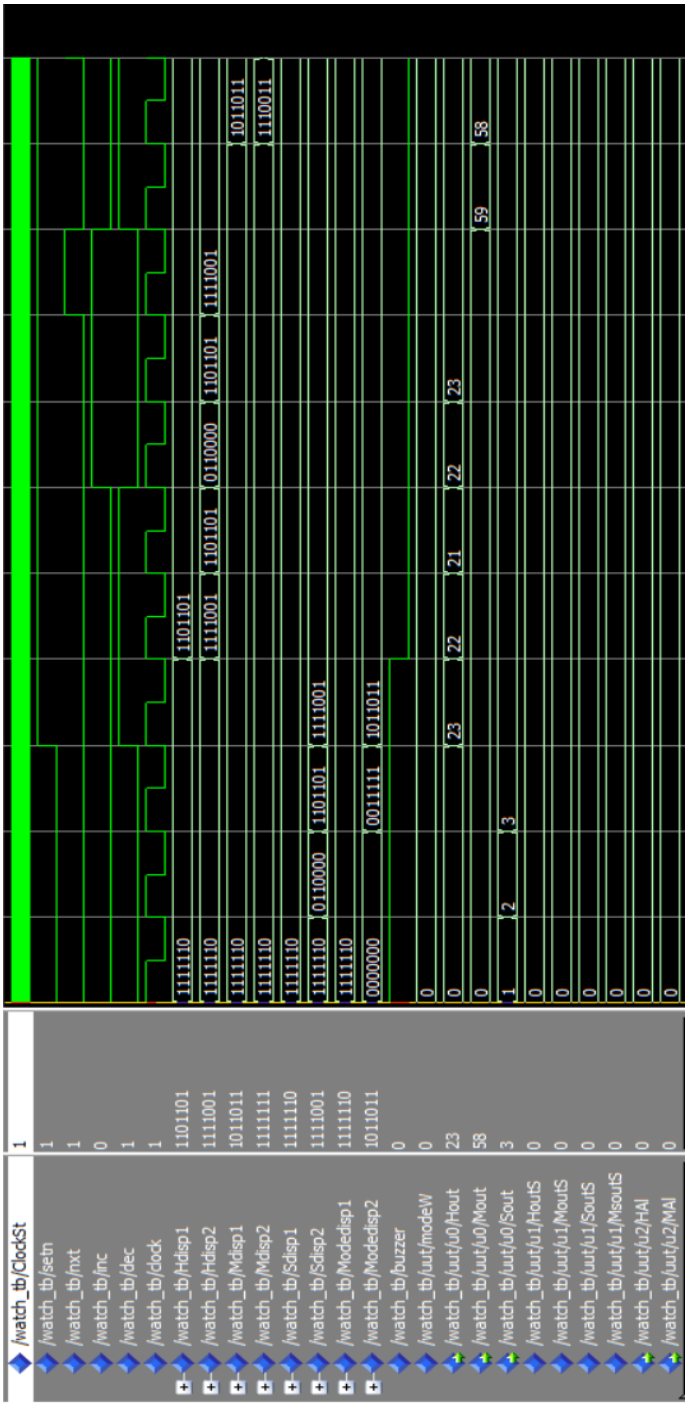

Figure 14: Runtime = 600ps
Mode set to change minutes: minutes set to 58.

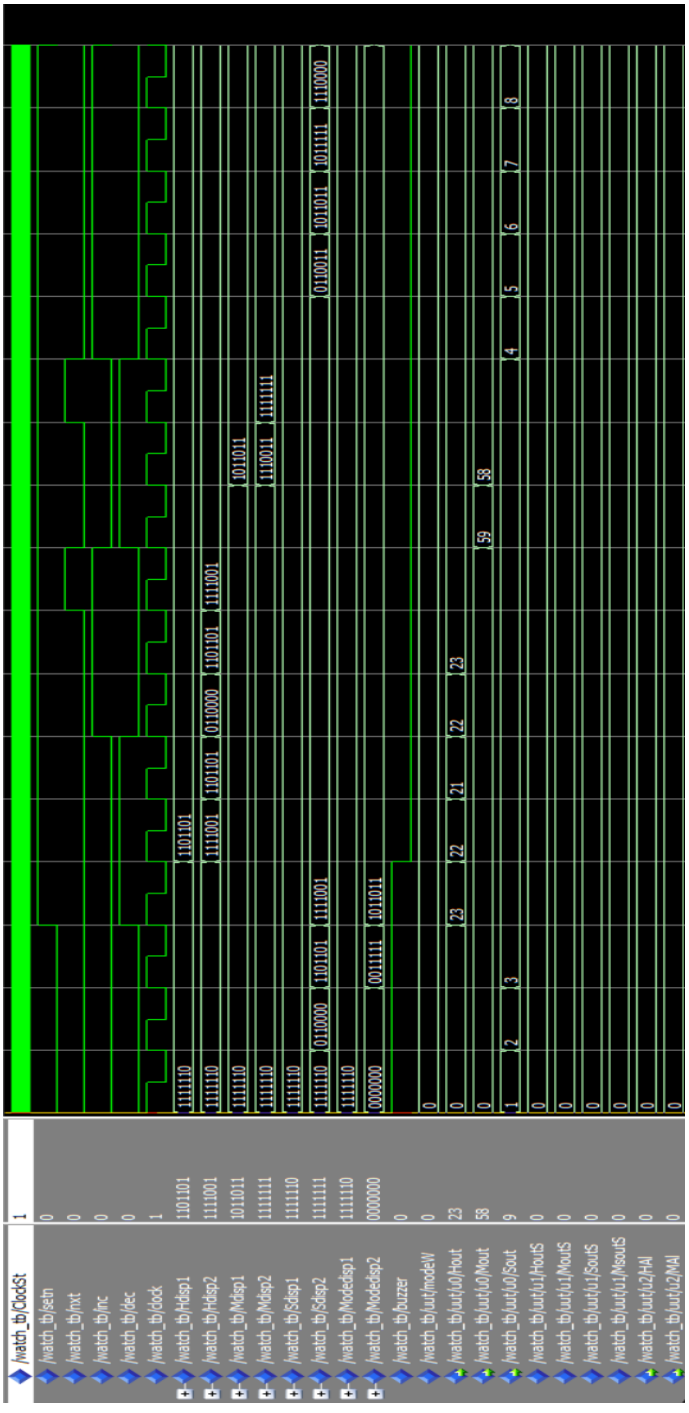

Figure 15: Runtime= 1200ps
Mode set to change seconds
Increments seconds by 5.

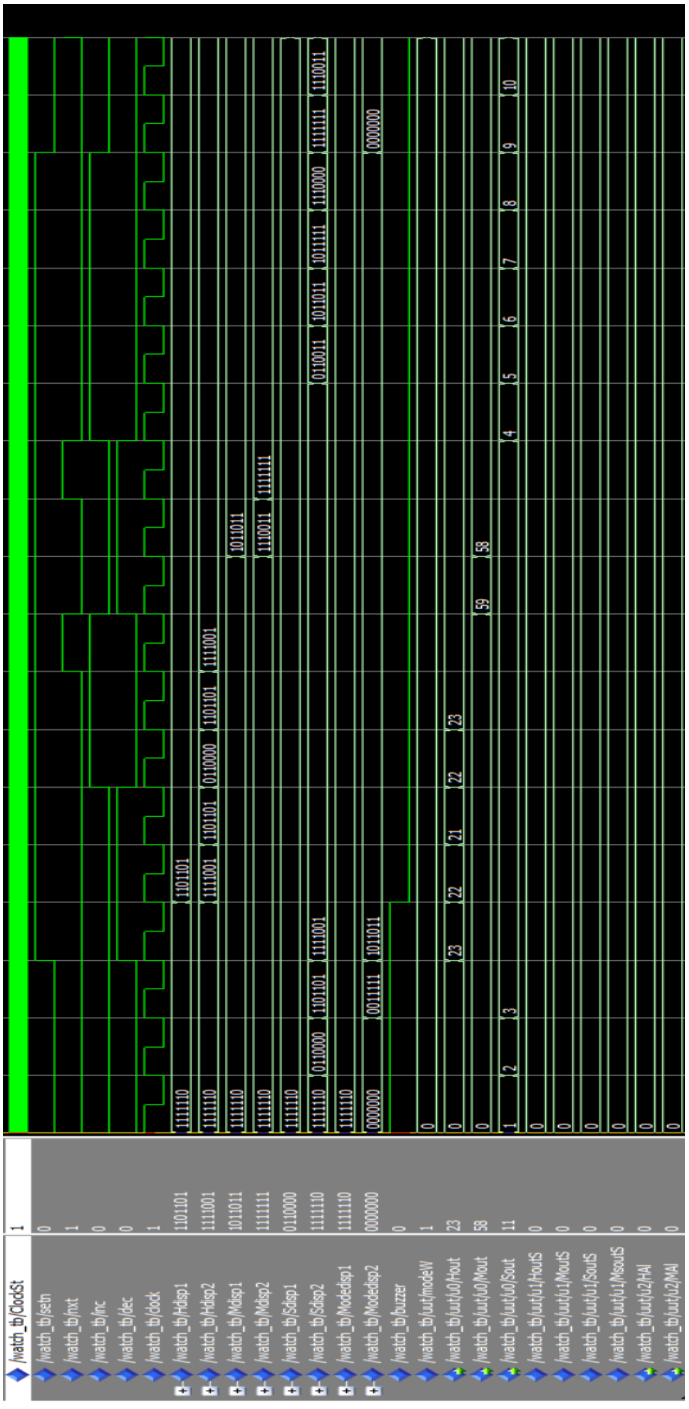

Figure 16: Runtime = 400ps
Runs the clock for 2 secs.

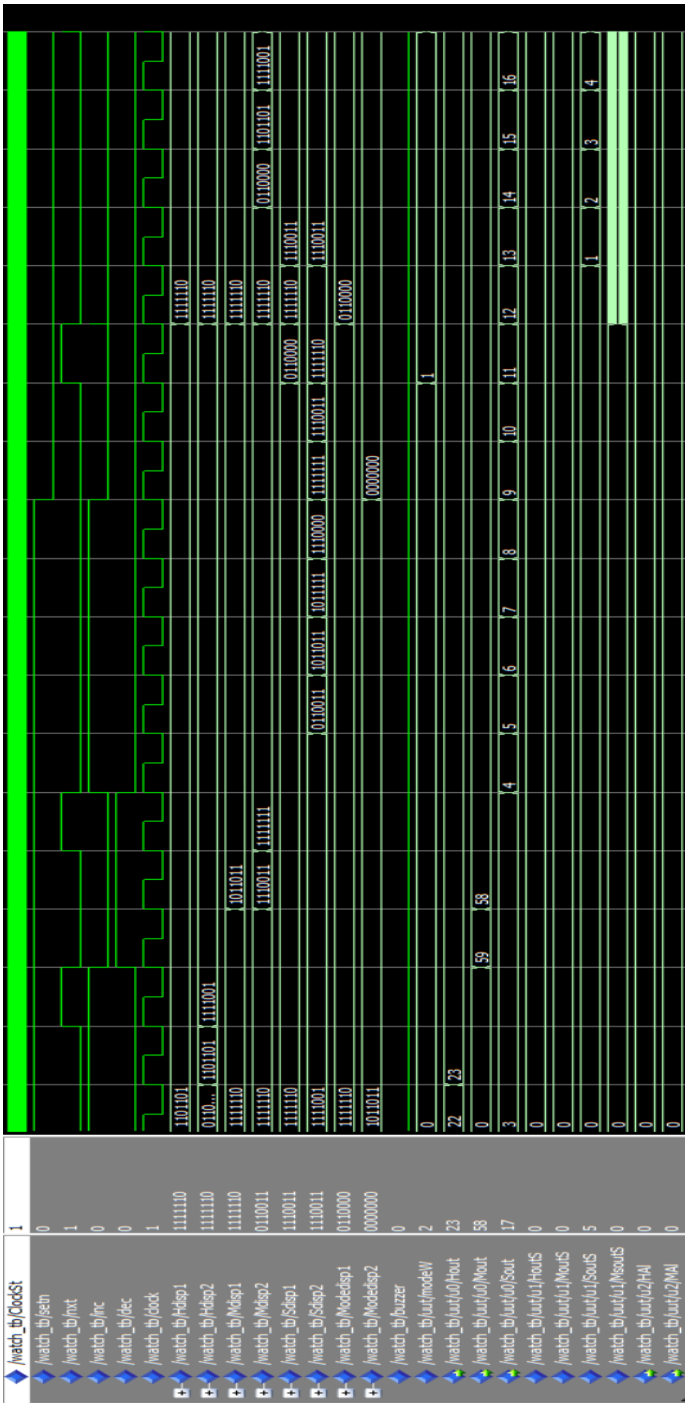

Figure 17: Runtime = 1200ps
Enters Stopwatch Mode, while the clock is running in the background

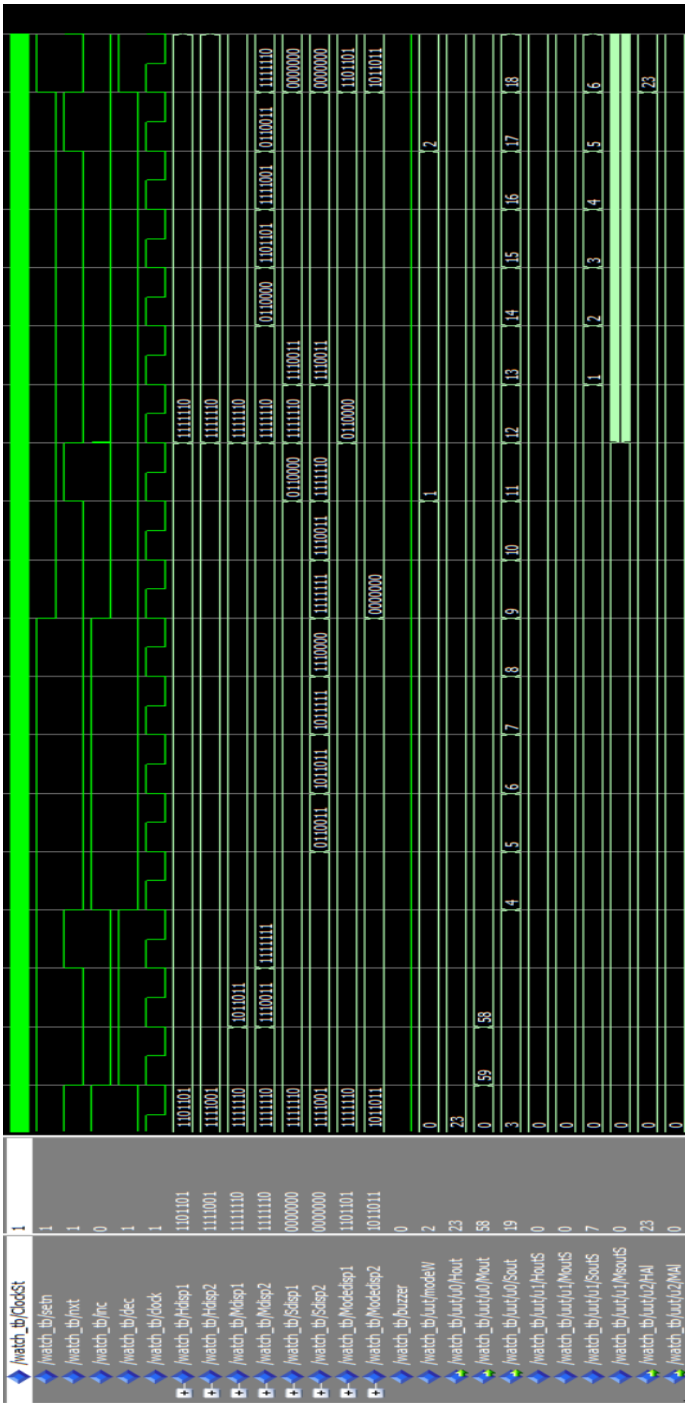

Figure 18: Runtime = 400ps. Mode : Setting an Alarm.
Hours set to 23.

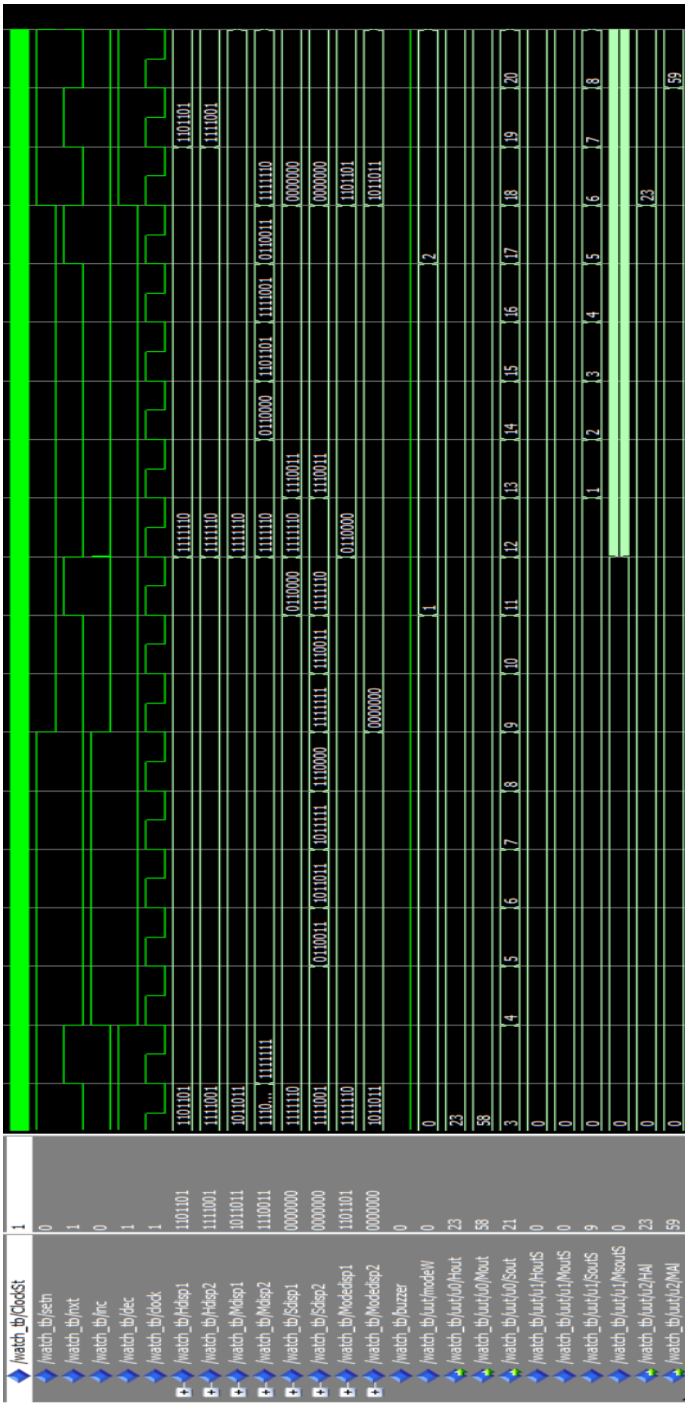

Figure 19: Runtime = 400ps. Mode : Setting an Alarm.
Minutes set to 59.

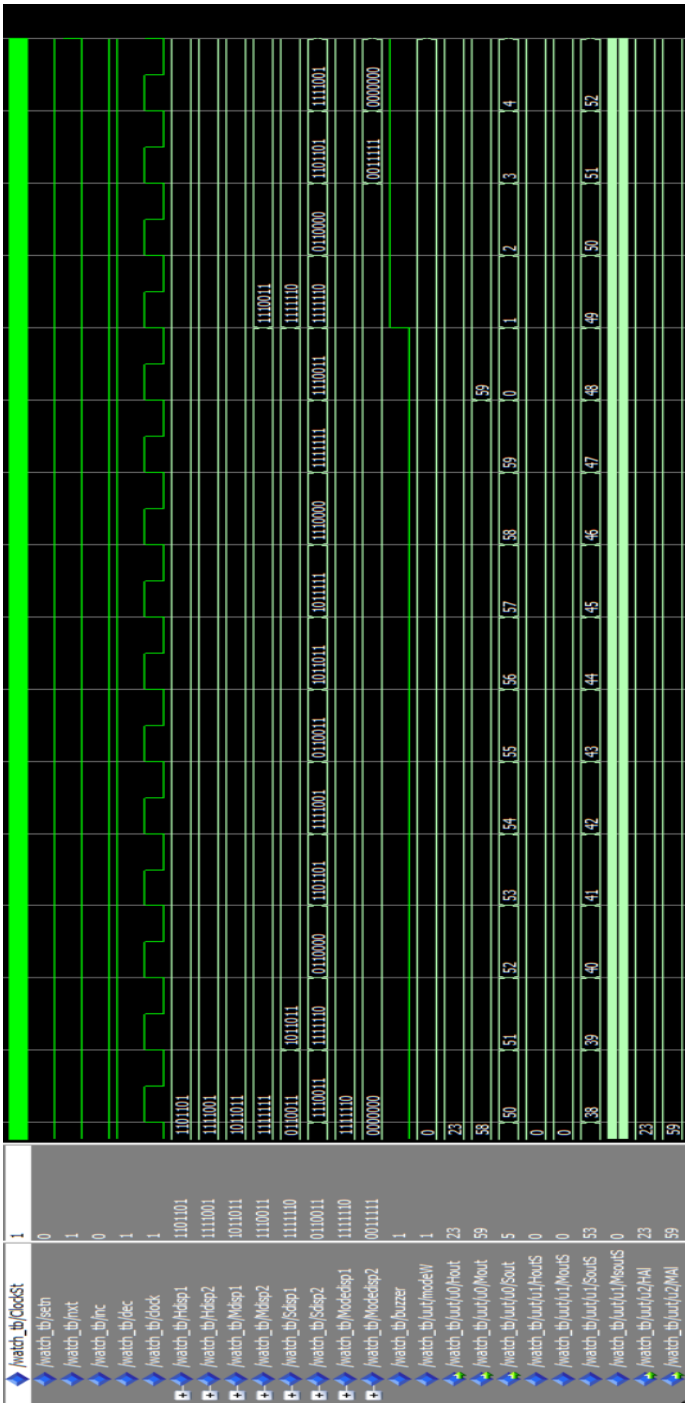

Figure 20: Runtime = 8800ps.
Normal Clock time reaches Set alarm time. Buzzer changes to 1, and Modedisp2 blinks.

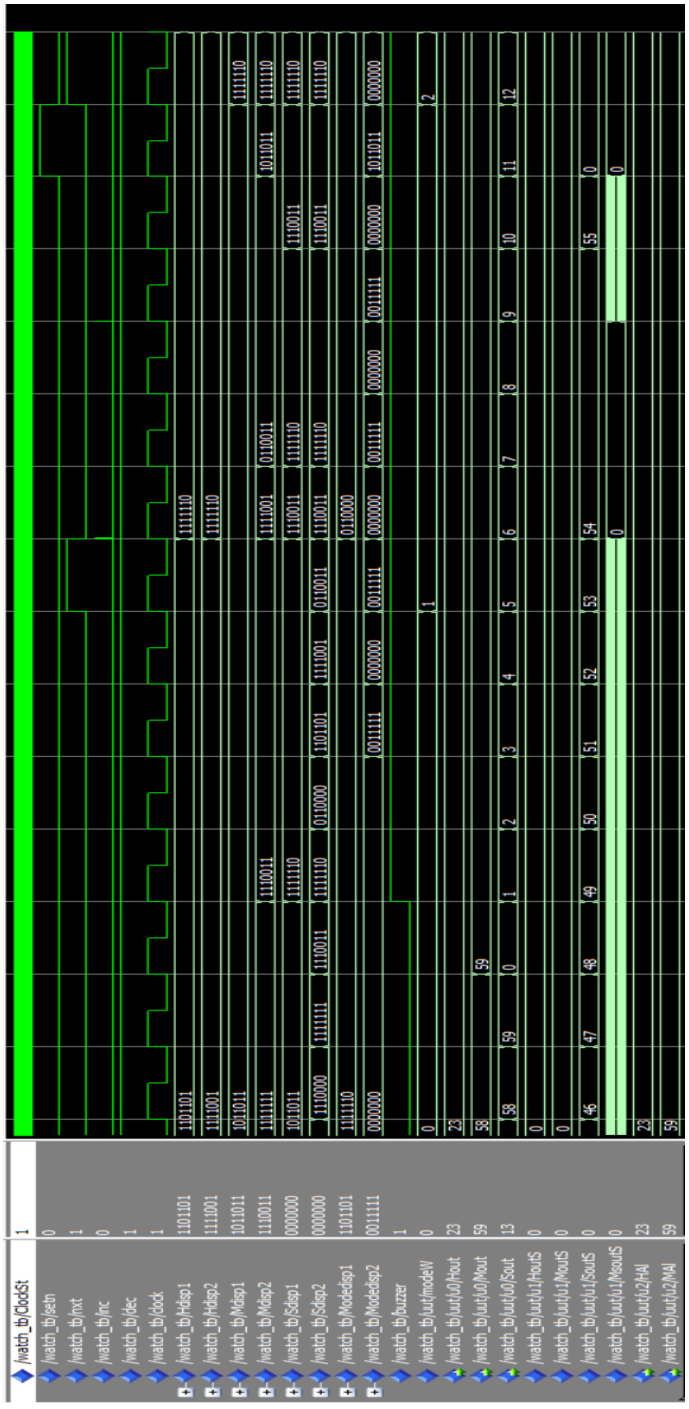

Figure 21: Runtime = 1200ps. Mode: Stopwatch. Stopwatch is paused for 3 secs and is then resumed for 2 secs.

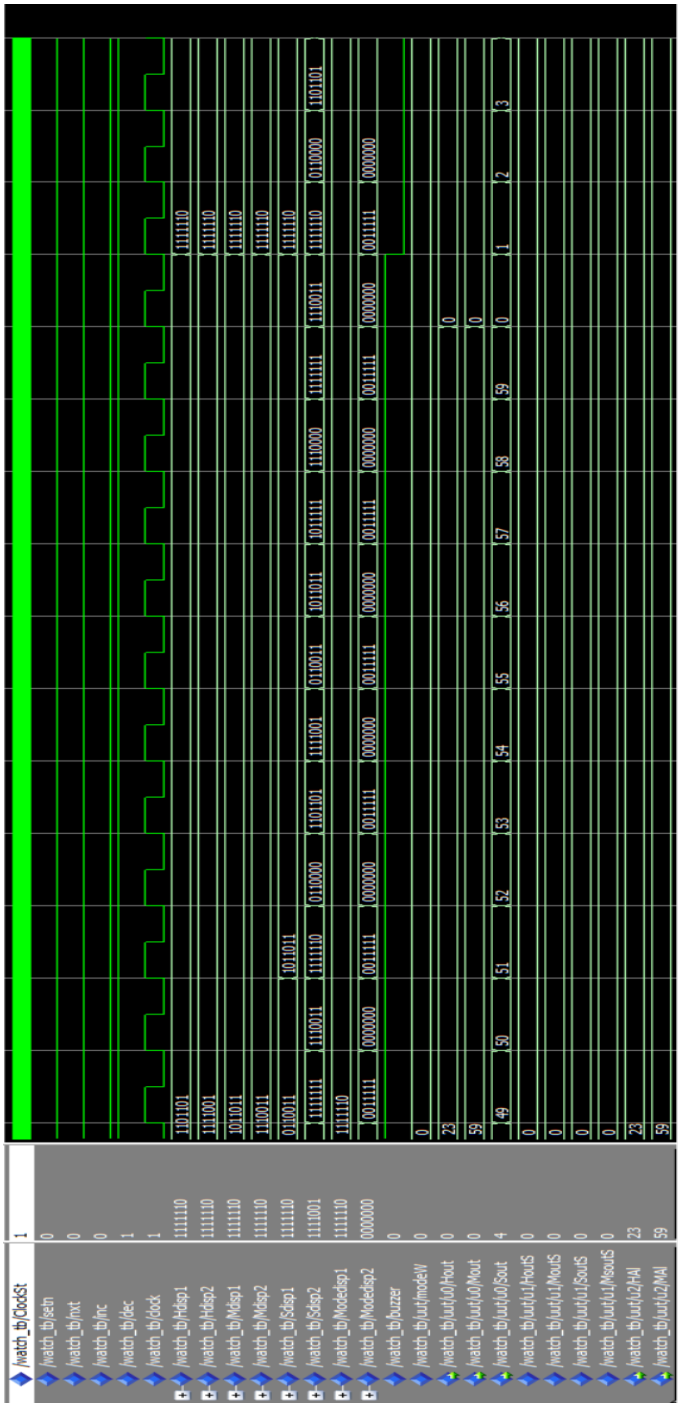

Figure 22: Runtime = 10400ps.
Clock resets to 0:0:0 after reaching 23:59:59.

1.6 Conclusion: Hence, We were able to Simulate a Digital Watch with Stopwatch and Alarm modes with a 7 - segment display output using VHDL coding.