Project Report

Design and Implementation of Digital Clock

Zhijie Lan 201990309, Zhongtian Wang 201991139

1. Introduction

This report gives a solution and relative discussion about the design and implementation of Digital Clocks using VHDL. The design will be shown step by step including the VHLD background, the initial block diagram, the state diagram, the implementation of codes and the test result.

Design Objectives:

a) Normal clock function: design a digital clock circuit with hour, minute, second displaying via LED digital tubes, and timing by 24 hours.

b) Stopwatch function: design a stopwatch that can accurately measure time, through the dynamic digital tube displaying minutes, seconds and milliseconds.

c) Alarm clock function: can be set in a time to ring alarm bells.

d) Calibration function: can manually adjust hours, minutes and seconds to calibrate the time.

Group Members and Contribution:

Group Member 1: Zhongtian Wang Student Number: 201991139
Group Member 2: Zhijie Lan Student Number: 201990309

Contribution of each member: The project content has been divided into some modules, and each member is responsible for the corresponding modules. Each member finished half workload of the whole project, including half of circuit design, half of VHDL programming, and half of report editing.

2. Background Information

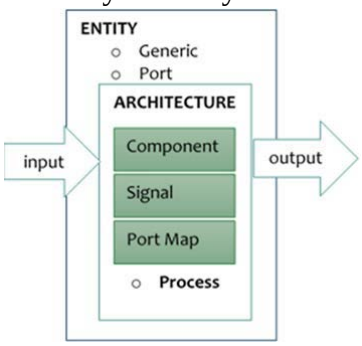
Digital Clock

The functions of a digital clock, though seemingly simple, are diverse and practical. To realize the function of a digital clock, it is necessary to utilize and combine a variety of basic devices of digital circuits, including using counters to realize timing, using synchronous counters to adjust the time, using registers to realize alarm clock function and so on.

Through the design of the experimental circuit, we can organically link the combinational logic circuit with the sequential logic circuit, which will deepen our understanding of the decoder, counter, and other functional components. Through the overall debugging, we can grasp the relationship between the modules, strengthening our ability to solve practical problems.

VHDL Overview

VHDL consists of an entity which can contain other entities as components of the top-level entity. Each entity is modeled by an entity declaration and an architecture body.



The entity structure of VHDL

Entity

An ENTITY represents a template for a hardware block. It describes just the outside view of a hardware module, namely its interface with other modules in terms of input and output signals. The inner operation of the entity is described by an ARCHITECTURE associated with it.

Architecture

An ARCHITECTURE describes how an ENTITY operates. It can describe an entity in a structural style, behavioural style or mixed style.

Dataflow Modelling

A dataflow architecture uses concurrent signal assignment statements. It describes a system in terms of how data flows through the system. Data dependencies in the description match those in a typical hardware implementation, directly implying a corresponding gate-level implementation.

Structural Modelling

A structural architecture uses component instantiation statements. The top-level design entity’s architecture describes the interconnection of lower-level design entities. Each lower-level design entity can be described as an interconnection of design entities at the next-lower level, and so on.

Behavioral Modelling

A behavioral architecture uses process statements to describe a system’s behavior or function in an algorithmic fashion. It consists of one or more process statements. Each process statement is a single concurrent statement that itself contains one or

more sequential statements.

Test Bench
To simulate a design containing a core, create a test bench file. The test bench should instantiate the top level module and should contain stimuli to drive the input ports of the design.

3. Methodology

We plan to use three buttons (Btn_switch, Btn_operate, and Btn_increase), which are used as input signals to change the mode, set the time, set the alarm, start/stop the stopwatch and so on, to implement all required functions of the digital clock. The design will use a logic control circuit that will realize four modes: mode 0, time display mode; mode 1, time setting mode; mode 2, accurate timing mode; mode 3, alarm setting mode.

Mode 0 will be implemented by counters. It needs to pay attention to time carry, and alarm signal rings at the hour time. Mode 1 will be realized by the synchronous setting of counters in mode 0. During the setting, the second should be timed normally to be close to the actual working mode of wristwatches. Mode 2 will be realized by counters, and high-frequency clocks will be adopted to realize accurate timing. Mode 3 will be realized by registers, and the alarm time will be controlled by keyboards. If it is the same as the counter in mode 0, the alarm will ring.

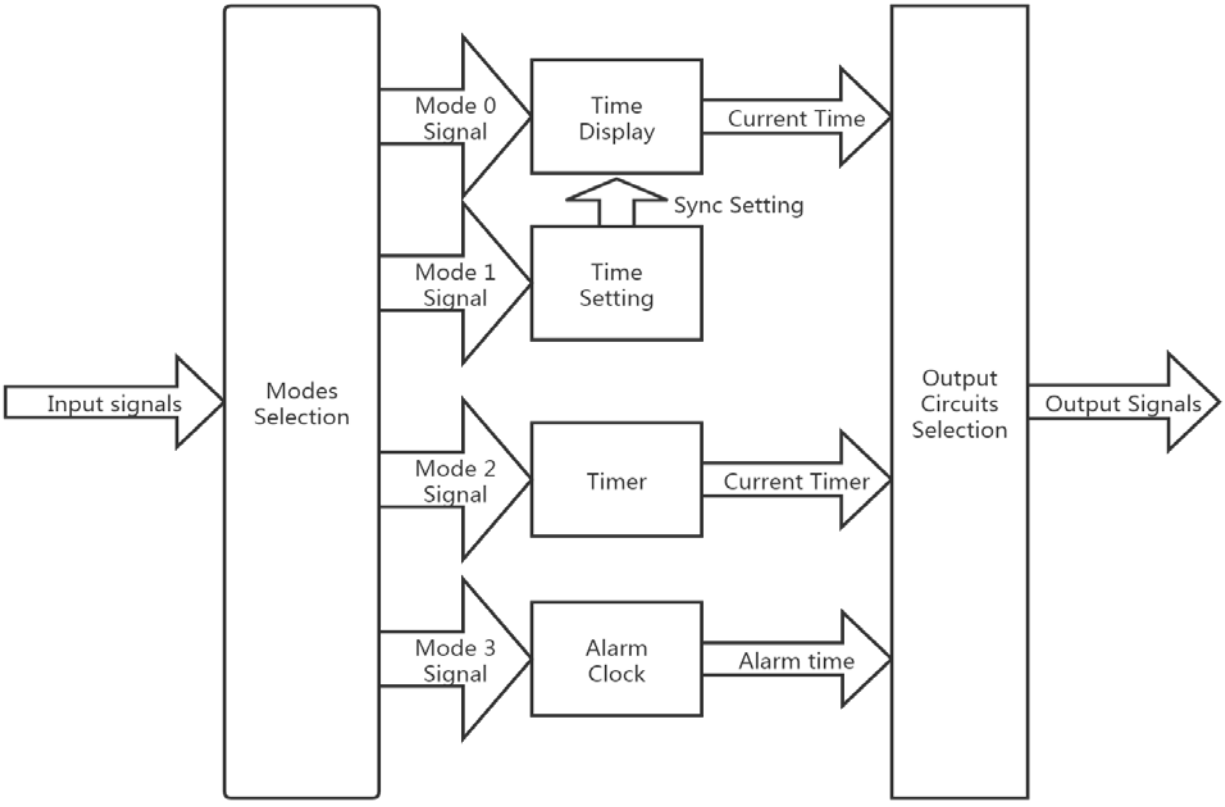


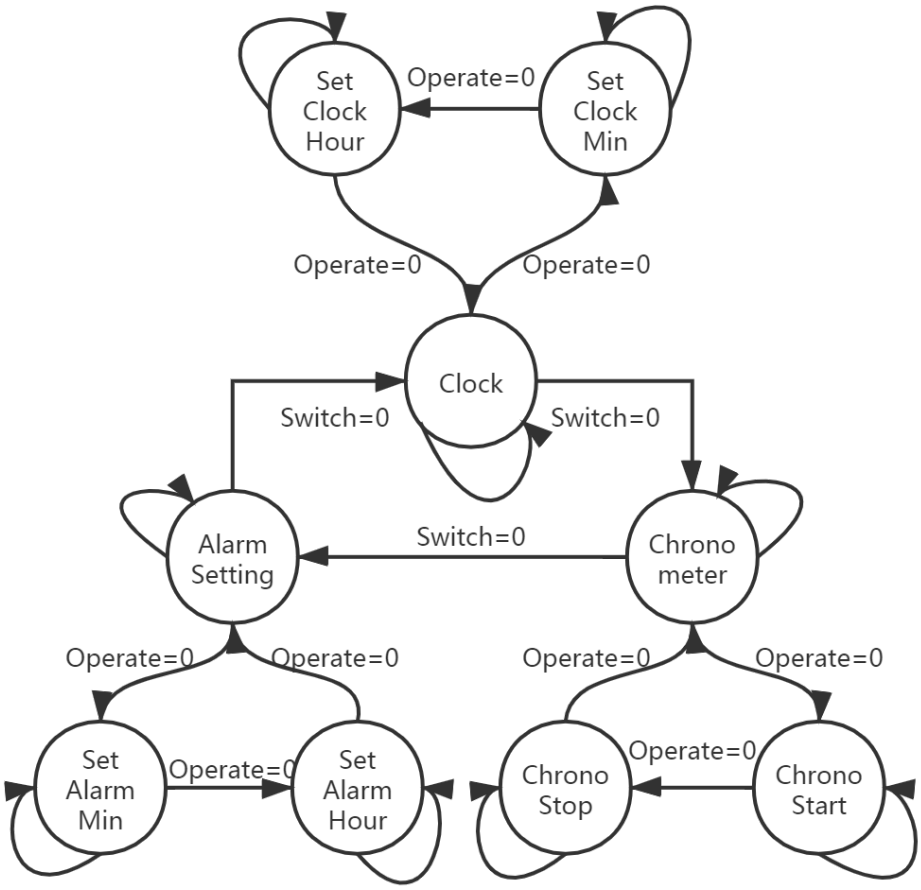
Diagram of control circuit

In particular, we will design and realize a frequency divider to control the working frequency of the circuit. There are many ways to realize the frequency divider. The most common way is to use the addition counter, that is to count the high frequency clock signal, and the corresponding frequency signal can be obtained at the output end of the counter.

The 7-segment LED will be used to display the outputs. After all the calculations inside the circuits, the data will finally be transformed to corresponding 7 bits code to be able to be shown as numbers on 7-segment LEDs.

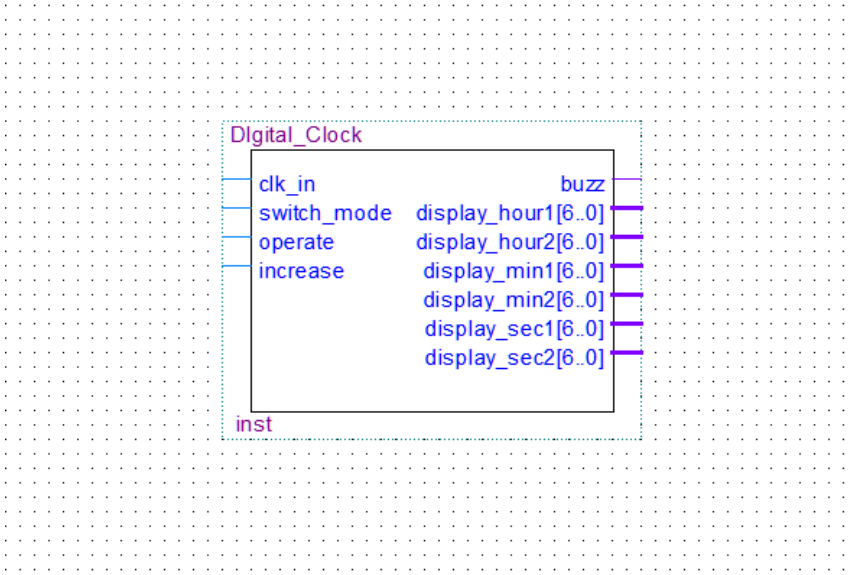
4. Design and Implementation

To implement those functions mentioned above, the finite state machine should be made. We decide to set 3 main state which is CLOCK, ALARM and CHRONO to represent 3 main functions (clock, alarm setting, stopwatch respectively). For each main state, there are two sub-states in order to achieve other operations.



State Diagram

From the state diagram, we can see that between the 3 main states, CLOCK, ALARM and CHRONO can be switched by the input signal SWITCH. In the CLOCK or ALARM state, the input signal OPERATE can switch the state form main state CLOCK or ALARM (normal display) to the state SET_MIN (select minutes, ready to be setting) and SET HOUR (select hours, ready to be setting). In the CHRONO state, the input signal OPERATE can switch the state form main state CHRONO (initial, ready to start) to the state START (start timing) and STOP (show the stop time).



Module of Digital Clock

The input buttons are SWITCH_MODE, OPERATE and INCREASE. The SWITCH_MODE is used to switch modes between 3 main states. The functions of OPERATE vary upon the mode selected by users. The INCREASE is to increase time when setting clock or alarm time. The CLK_IN signal will input 50MHz clock signal. The outputs should directly be connected to 7-segment LEDs since all the corresponding calculations and transformations are done inside the module.

Code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
--Digital Clock project

--201991139 WANG ZHONGTIAN
--201990309 LAN ZHIJIE

entity DIgital_Clock is
port (
                clk_in                  : in std_logic;
```

```
                switch_mode             : in std_logic := '1';--switch between clock,chronometer and alarm
                operate                 : in std_logic := '1';--select min or hour when in clock or alarm.
start/stop when in chronometer
                increase        : in std_logic := '1';--time + 1 when select min or hour
                buzz                            : out std_logic;-- buzzing when clock time is same with
alarm time
                display_hour1   : out std_logic_vector(6 downto 0);--display on 6 bits 7-segement LED
                display_hour2   : out std_logic_vector(6 downto 0);
                display_min1    : out std_logic_vector(6 downto 0);
                display_min2    : out std_logic_vector(6 downto 0);
                display_sec1    : out std_logic_vector(6 downto 0);
                display_sec2    : out std_logic_vector(6 downto 0)
        );
end DIgital_Clock;

architecture design of DIgital_Clock is
        type states is (clock, clk_min, clk_hour, chrono, chr_start, chr_stop, alarm, alm_min, alm_hour);
        --state signal
        signal state                                    : states := clock;

        --time in clock state
        signal timing_sec1              : integer := 0;
        signal timing_sec2              : integer := 0;
        signal timing_min1              : integer := 0;
        signal timing_min2              : integer := 0;
        signal timing_hour1             : integer := 0;
        signal timing_hour2             : integer := 0;

        --time in setting clock or alarm
        signal timing_set_alarm_min1            : integer := 0;
        signal timing_set_alarm_min2            : integer := 0;
        signal timing_set_alarm_hour1           : integer := 0;
        signal timing_set_alarm_hour2           : integer := 0;
        signal timing_set_clk_min1                      : integer := 0;
        signal timing_set_clk_min2              : integer := 0;
        signal timing_set_clk_hour1             : integer := 0;
        signal timing_set_clk_hour2             : integer := 0;

        --time in chronometer
        signal timing_chrono_mili1              : integer := 0;
        signal timing_chrono_mili2              : integer := 0;
        signal timing_chrono_sec1               : integer := 0;
        signal timing_chrono_sec2               : integer := 0;
        signal timing_chrono_min1               : integer := 0;
        signal timing_chrono_min2               : integer := 0;

        --setting min or hour of clock time
        signal set_min                  : std_logic := '0';
```

```
        signal set_hour                 : std_logic := '0';

        --setting min or hour of alarm time
        signal set_alm_min              : std_logic := '0';
        signal set_alm_hour  : std_logic := '0';

        --chronometer start signal
        signal chrono_start    : std_logic := '0';

        -- clock display
        signal display_clk_sec1                         : std_logic_vector(6 downto 0);
        signal display_clk_sec2                         : std_logic_vector(6 downto 0);
        signal display_clk_min1                         : std_logic_vector(6 downto 0);
        signal display_clk_min2                         : std_logic_vector(6 downto 0);
        signal display_clk_hour1                : std_logic_vector(6 downto 0);
        signal display_clk_hour2                : std_logic_vector(6 downto 0);

        --setting clock display
        signal display_set_clk_min1             : std_logic_vector(6 downto 0);
        signal display_set_clk_min2             : std_logic_vector(6 downto 0);
        signal display_set_clk_hour1            : std_logic_vector(6 downto 0);
        signal display_set_clk_hour2            : std_logic_vector(6 downto 0);

        --chronometer display
        signal display_chrono_mili1             : std_logic_vector(6 downto 0);
        signal display_chrono_mili2             : std_logic_vector(6 downto 0);
        signal display_chrono_sec1                      : std_logic_vector(6 downto 0);
        signal display_chrono_sec2                      : std_logic_vector(6 downto 0);
        signal display_chrono_min1                      : std_logic_vector(6 downto 0);
        signal display_chrono_min2                      : std_logic_vector(6 downto 0);

        --setting alarm display
        signal display_set_alarm_min1                   : std_logic_vector(6 downto 0);
        signal display_set_alarm_min2                   : std_logic_vector(6 downto 0);
        signal display_set_alarm_hour1          : std_logic_vector(6 downto 0);
        signal display_set_alarm_hour2          : std_logic_vector(6 downto 0);

        --alarm and clock
        signal clk                              : std_logic :='0';

        --high frequency clock
        signal chronoclk                : std_logic :='0';

begin

        --states
        States_process: process(chronoclk)
        begin
                if rising_edge(chronoclk) then
```

```
case state is
        when clock =>
                if switch_mode = '0' then
                        state <= chrono;
                elsif operate <= '0' then
                        state <= clk_min;
                else
                        state <= clock;
                end if;

        when clk_min =>
                if operate = '0' then
                        state <= clk_hour;
                else
                        state <= clk_min;
                end if;

        when clk_hour =>
                if operate = '0' then
                        state <= clock;
                else
                        state <= clk_hour;
                end if;

        when chrono =>
                if switch_mode = '0' then
                        state <= alarm;
                elsif operate = '0' then
                        state <= chr_start;
                else
                        state <= chrono;
                end if;

        when chr_start =>
                if operate = '0' then
                        state <= chr_stop;
                else
                        state <= chr_start;
                end if;

        when chr_stop =>
                if operate = '0' then
                        state <= chrono;
                else
                        state <= chr_stop;
                end if;

        when alarm =>
                if switch_mode = '0' then
```

```
                                        state <= clock;
                                elsif operate = '0' then
                                        state <= alm_min;
                                else
                                        state <= alarm;
                                end if;

                        when alm_min =>
                                if operate = '0' then
                                        state <= alm_hour;
                                else
                                        state <= alm_min;
                                end if;

                        when alm_hour =>
                                if operate = '0' then
                                        state <= alarm;
                                else
                                        state <= alm_hour;
                                end if;
                end case;
        end if;
end process states_process;

--state behavior
displays: process(state, chronoclk)
begin
        if rising_edge(chronoclk) then
                case state is
                        when clock => --display clock time
                                display_sec1 <= display_clk_sec1;
                                display_sec2 <= display_clk_sec2;
                                display_min1 <= display_clk_min1;
                                display_min2 <= display_clk_min2;
                                display_hour1 <= display_clk_hour1;
                                display_hour2 <= display_clk_hour2;
                                set_min <= '0';
                                set_hour <= '0';
                        when clk_min => -- display min set
                                display_min1 <= display_set_clk_min1;
                                display_min2 <= display_set_clk_min2;
                                display_sec1 <= display_clk_sec1;
                                display_sec2 <= display_clk_sec2;
                                display_hour1 <= display_clk_hour1;
                                display_hour2 <= display_clk_hour2;
                                set_min <= '1';
                                set_hour <= '0';
                        when clk_hour => -- display hour set
                                display_hour1 <= display_set_clk_hour1;
```

```
                display_hour2 <= display_set_clk_hour2;
                display_sec1 <= display_clk_sec1;
                display_sec2 <= display_clk_sec2;
                display_min1 <= display_clk_min1;
                display_min2 <= display_clk_min2;
                set_hour <= '1';
                set_min <= '0';
            when chrono => -- display chronometer
                display_sec1 <= "1000000";
                display_sec2 <= "1000000";
                display_min1 <= "1000000";
                display_min2 <= "1000000";
                display_hour1 <= "1000000";
                display_hour2 <= "1000000";
            when chr_start => -- start timing
                display_sec1 <= display_chrono_mili1;
                display_sec2 <= display_chrono_mili2;
                display_min1 <= display_chrono_sec1;
                display_min2 <= display_chrono_sec2;
                display_hour1 <= display_chrono_min1;
                display_hour2 <= display_chrono_min2;
                chrono_start <= '1';
            when chr_stop => --show stop time
                display_sec1 <= display_chrono_mili1;
                display_sec2 <= display_chrono_mili2;
                display_min1 <= display_chrono_sec1;
                display_min2 <= display_chrono_sec2;
                display_hour1 <= display_chrono_min1;
                display_hour2 <= display_chrono_min2;
                chrono_start <= '0';
            when alarm => -- display alarm time
                display_sec1 <= "1000000";
                display_sec2 <= "1000000";
                display_min1 <= display_set_alarm_min1;
                display_min2 <= display_set_alarm_min2;
                display_hour1 <= display_set_alarm_hour1;
                display_hour2 <= display_set_alarm_hour2;
                set_alm_min <= '0';
                set_alm_hour <= '0';
            when alm_min => --display set alarm
                display_sec1 <= "1000000";
                display_sec2 <= "1000000";
                display_min1 <= display_set_alarm_min1;
                display_min2 <= display_set_alarm_min2;
                display_hour1 <= display_set_alarm_hour1;
                display_hour2 <= display_set_alarm_hour2;
                set_alm_min <= '1';
                set_alm_hour <= '0';
            when alm_hour => -- display set alarm
```

```
                                        display_sec1 <= "1000000";
                                        display_sec2 <= "1000000";
                                        display_min1 <= display_set_alarm_min1;
                                        display_min2 <= display_set_alarm_min2;
                                        display_hour1 <= display_set_alarm_hour1;
                                        display_hour2 <= display_set_alarm_hour2;
                                        set_alm_hour <= '1';
                                        set_alm_min <= '0';
                        end case;
                end if;
        end process displays;

--clock and alarm
        timing: process(clk) --every 1 second.
        variable sec,min,hour                   : integer := 0;
        variable alarm_hour                     : integer := 0;
        variable alarm_min                      : integer := 0;
        begin
                if(clk'event and clk='1') then
                        if set_min = '1' then
                                if increase = '0' then
                                        min := min + 1;
                                        timing_set_clk_min1 <= min/10;
                                        timing_set_clk_min2 <= min rem 10;
                                end if;
                                timing_min1 <= min/10;
                                timing_min2 <= min rem 10;
                        elsif set_hour = '1' then
                                if increase = '0' then
                                        hour := hour + 1;
                                        timing_set_clk_hour1 <= hour/10;
                                        timing_set_clk_hour2 <= hour rem 10;
                                end if;
                                timing_hour1 <= hour/10;
                                timing_hour2 <= hour rem 10;
                        end if;

                        if min = alarm_min and hour = alarm_hour then
                                buzz <= '1';
                        else
                                buzz <= '0';
                        end if;

                        sec := sec + 1;
                        if(sec > 59) then
                                sec             := 0;
                                min := min + 1;
                                if(min > 59) then
                                        hour            := hour + 1;
```

```
                                        min         := 0;
                                        if(hour > 23) then
                                                hour := 0;
                                        end if;
                                end if;
                        end if;
                        timing_sec1 <= sec/10;
                        timing_sec2 <= sec rem 10;
                        timing_min1 <= min/10;
                        timing_min2 <= min rem 10;
                        timing_hour1 <= hour/10;
                        timing_hour2 <= hour rem 10;
                end if;
                if(clk'event and clk='0') then
                        if set_alm_min = '1' then
                                if increase = '0' then
                                        alarm_min := alarm_min + 1;
                                        timing_set_alarm_min1 <= alarm_min/10;
                                        timing_set_alarm_min2 <= alarm_min rem 10;
                                end if;
                        elsif set_alm_hour = '1' then
                                if increase = '0' then
                                        alarm_hour := alarm_hour + 1;
                                        timing_set_alarm_hour1 <= alarm_hour/10;
                                        timing_set_alarm_hour2 <= alarm_hour rem 10;
                                end if;
                        end if;
                end if;
end process timing;

--chronometer
chrono_timing: process(chronoclk, clk, chrono_start)
variable mili,sec,min           : integer := 0;
begin
                if chrono_start = '1' then
                        if(chronoclk'event and chronoclk='1') then
                                mili := mili + 1;
                                if(mili > 99) then
                                        mili        := 0;
                                end if;
                                timing_chrono_mili1 <= mili/10;
                                timing_chrono_mili2 <= mili rem 10;
                        end if;
                        if(clk'event and clk='1') then
                                sec         := sec + 1;
                                if(sec > 59) then
                                        sec         := 0;
                                        min         := min + 1;
                                end if;
```

```
                        timing_chrono_sec1 <= sec/10;
                        timing_chrono_sec2 <= sec rem 10;
                        timing_chrono_min1 <= min/10;
                        timing_chrono_min2 <= min rem 10;
                end if;
        else
                mili                    := mili;
                sec                     := sec;
                min                     := min;
                timing_chrono_mili1             <= mili/10;
                timing_chrono_mili2             <= mili rem 10;
                timing_chrono_sec1              <= sec/10;
                timing_chrono_sec2              <= sec rem 10;
                timing_chrono_min1              <= min/10;
                timing_chrono_min2              <= min rem 10;
                if operate = '0' then
                        mili                    := 0;
                        sec                     := 0;
                        min                     := 0;
                        timing_chrono_mili1             <= mili/10;
                        timing_chrono_mili2             <= mili rem 10;
                        timing_chrono_sec1              <= sec/10;
                        timing_chrono_sec2              <= sec rem 10;
                        timing_chrono_min1              <= min/10;
                        timing_chrono_min2              <= min rem 10;
                end if;
        end if;
end process chrono_timing;

--nomal clk.
--produces 1 Hz clock from 50MHz.
clocking: process(clk_in)
variable count : integer := 1;
begin
        if(clk_in'event and clk_in = '1') then
                count := count + 1;
        --              if(count = 25000000) then
        -- set the count as a small number for testing
                if(count = 10) then
                        clk <= not clk;
                        count := 1;
                end if;
        end if;
end process clocking;

--high frequency clk.
--produces 1000 Hz clock from 50MHz.
chrono_clocking: process(clk_in)
variable count : integer := 1;
```

```
begin
        if(clk_in'event and clk_in='1') then
                count := count + 1;
        --      if(count = 25000) then
        -- set the count as a small number for testing
                if(count = 2) then
                        chronoclk <= not chronoclk;
                        count := 1;
                end if;
        end if;
end process chrono_clocking;


--display output:
WITH timing_sec1 SELECT
        display_clk_sec1 <=  "1000000" WHEN 0,
                             "1111001" WHEN 1,
                             "0100100" WHEN 2,
                             "0110000" WHEN 3,
                             "0011001" WHEN 4,
                             "0010010" WHEN 5,
                             "0000010" WHEN 6,
                             "1111000" WHEN 7,
                             "0000000" WHEN 8,
                             "0010000" WHEN 9,
                             "0000000" WHEN OTHERS;
WITH timing_sec2 SELECT
        display_clk_sec2 <=  "1000000" WHEN 0,
                             "1111001" WHEN 1,
                             "0100100" WHEN 2,
                             "0110000" WHEN 3,
                             "0011001" WHEN 4,
                             "0010010" WHEN 5,
                             "0000010" WHEN 6,
                             "1111000" WHEN 7,
                             "0000000" WHEN 8,
                             "0010000" WHEN 9,
                             "0000000" WHEN OTHERS;
WITH timing_min1 SELECT
        display_clk_min1 <=  "1000000" WHEN 0,
                             "1111001" WHEN 1,
                             "0100100" WHEN 2,
                             "0110000" WHEN 3,
                             "0011001" WHEN 4,
                             "0010010" WHEN 5,
                             "0000010" WHEN 6,
                             "1111000" WHEN 7,
                             "0000000" WHEN 8,
                             "0010000" WHEN 9,
```

```
                                    "0000000" WHEN OTHERS;
        WITH timing_min2 SELECT
                display_clk_min2 <=  "1000000" WHEN 0,
                                    "1111001" WHEN 1,
                                    "0100100" WHEN 2,
                                    "0110000" WHEN 3,
                                    "0011001" WHEN 4,
                                    "0010010" WHEN 5,
                                    "0000010" WHEN 6,
                                    "1111000" WHEN 7,
                                    "0000000" WHEN 8,
                                    "0010000" WHEN 9,
                                    "0000000" WHEN OTHERS;
        WITH timing_hour1 SELECT
        display_clk_hour1 <=        "1000000" WHEN 0,
                                    "1111001" WHEN 1,
                                    "0100100" WHEN 2,
                                    "0110000" WHEN 3,
                                    "0011001" WHEN 4,
                                    "0010010" WHEN 5,
                                    "0000010" WHEN 6,
                                    "1111000" WHEN 7,
                                    "0000000" WHEN 8,
                                    "0010000" WHEN 9,
                                    "0000000" WHEN OTHERS;
        WITH timing_hour2 SELECT
        display_clk_hour2 <=        "1000000" WHEN 0,
                                    "1111001" WHEN 1,
                                    "0100100" WHEN 2,
                                    "0110000" WHEN 3,
                                    "0011001" WHEN 4,
                                    "0010010" WHEN 5,
                                    "0000010" WHEN 6,
                                    "1111000" WHEN 7,
                                    "0000000" WHEN 8,
                                    "0010000" WHEN 9,
                                    "0000000" WHEN OTHERS;
        WITH timing_set_alarm_min1 SELECT
display_set_alarm_min1 <=    "1000000" WHEN 0,
                                    "1111001" WHEN 1,
                                    "0100100" WHEN 2,
                                    "0110000" WHEN 3,
                                    "0011001" WHEN 4,
                                    "0010010" WHEN 5,
                                    "0000010" WHEN 6,
                                    "1111000" WHEN 7,
                                    "0000000" WHEN 8,
                                    "0010000" WHEN 9,
                                    "0000000" WHEN OTHERS;
```

```
        WITH timing_set_alarm_min2 SELECT
display_set_alarm_min2 <=       "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
        WITH timing_set_alarm_hour1 SELECT
display_set_alarm_hour1 <=      "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
        WITH timing_set_alarm_hour2 SELECT
display_set_alarm_hour2 <=      "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
        WITH timing_set_clk_min1 SELECT
display_set_clk_min1 <=         "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
        WITH timing_set_clk_min2 SELECT
```

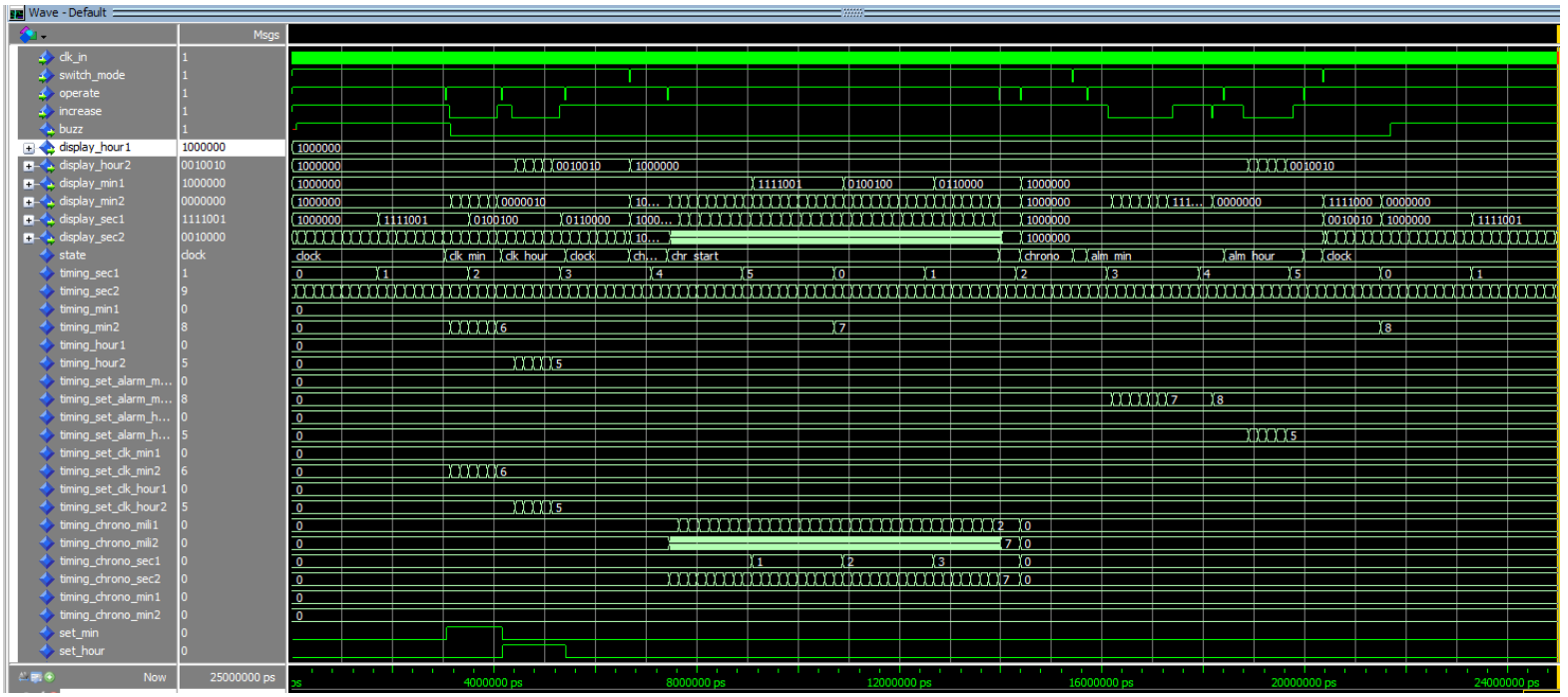
```
display_set_clk_min2 <=          "1000000" WHEN 0,
                                 "1111001" WHEN 1,
                                 "0100100" WHEN 2,
                                 "0110000" WHEN 3,
                                 "0011001" WHEN 4,
                                 "0010010" WHEN 5,
                                 "0000010" WHEN 6,
                                 "1111000" WHEN 7,
                                 "0000000" WHEN 8,
                                 "0010000" WHEN 9,
                                 "0000000" WHEN OTHERS;
        WITH timing_set_clk_hour1 SELECT
display_set_clk_hour1 <=         "1000000" WHEN 0,
                                 "1111001" WHEN 1,
                                 "0100100" WHEN 2,
                                 "0110000" WHEN 3,
                                 "0011001" WHEN 4,
                                 "0010010" WHEN 5,
                                 "0000010" WHEN 6,
                                 "1111000" WHEN 7,
                                 "0000000" WHEN 8,
                                 "0010000" WHEN 9,
                                 "0000000" WHEN OTHERS;
        WITH timing_set_clk_hour2 SELECT
display_set_clk_hour2 <=         "1000000" WHEN 0,
                                 "1111001" WHEN 1,
                                 "0100100" WHEN 2,
                                 "0110000" WHEN 3,
                                 "0011001" WHEN 4,
                                 "0010010" WHEN 5,
                                 "0000010" WHEN 6,
                                 "1111000" WHEN 7,
                                 "0000000" WHEN 8,
                                 "0010000" WHEN 9,
                                 "0000000" WHEN OTHERS;
        WITH timing_chrono_mili1 SELECT
display_chrono_mili1 <=          "1000000" WHEN 0,
                                 "1111001" WHEN 1,
                                 "0100100" WHEN 2,
                                 "0110000" WHEN 3,
                                 "0011001" WHEN 4,
                                 "0010010" WHEN 5,
                                 "0000010" WHEN 6,
                                 "1111000" WHEN 7,
                                 "0000000" WHEN 8,
                                 "0010000" WHEN 9,
                                 "0000000" WHEN OTHERS;
        WITH timing_chrono_mili2 SELECT
        display_chrono_mili2 <=          "1000000" WHEN 0,
```

```
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
WITH timing_chrono_sec1 SELECT
display_chrono_sec1 <=          "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
WITH timing_chrono_sec2 SELECT
display_chrono_sec2 <=          "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
WITH timing_chrono_min1 SELECT
display_chrono_min1 <=          "1000000" WHEN 0,
                                "1111001" WHEN 1,
                                "0100100" WHEN 2,
                                "0110000" WHEN 3,
                                "0011001" WHEN 4,
                                "0010010" WHEN 5,
                                "0000010" WHEN 6,
                                "1111000" WHEN 7,
                                "0000000" WHEN 8,
                                "0010000" WHEN 9,
                                "0000000" WHEN OTHERS;
WITH timing_chrono_min2 SELECT
display_chrono_min2 <=          "1000000" WHEN 0,
                                "1111001" WHEN 1,
```

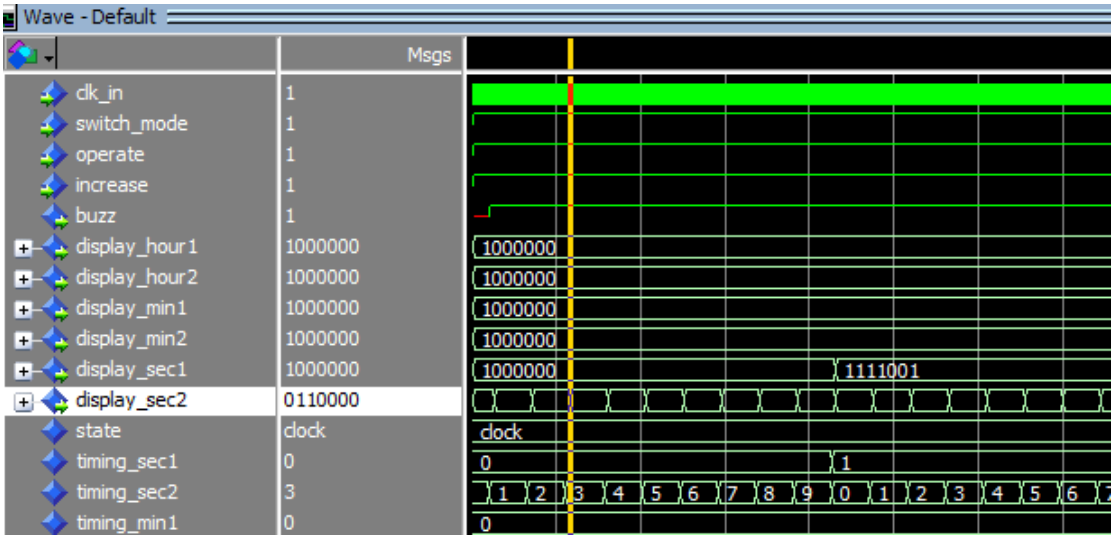
```
"0100100" WHEN 2
"0110000" WHEN 3,
"0011001" WHEN 4,
"0010010" WHEN 5,
"0000010" WHEN 6,
"1111000" WHEN 7,
"0000000" WHEN 8,
"0010000" WHEN 9,
"0000000" WHEN OTHERS;
```

```
end design;
```

5. Test Result

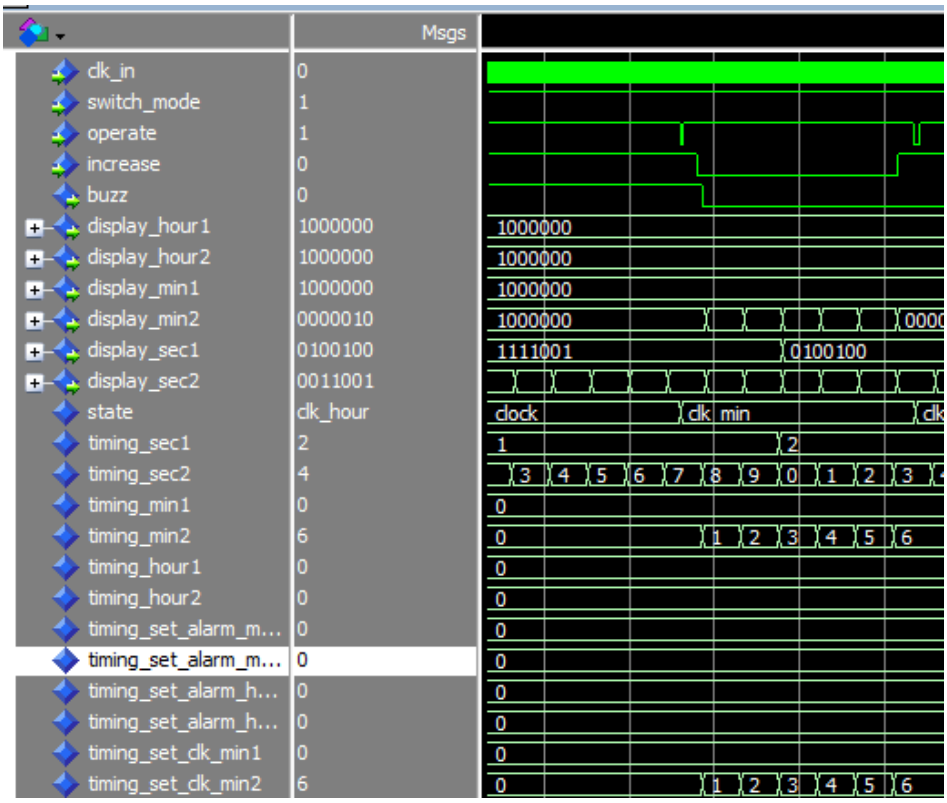


This is our complete screenshot, in this part we will analyze and explain each part of this screenshot. This screenshot contains all status displays. In order to simulate in a short time, we edited the count number of the clock divider in the code.

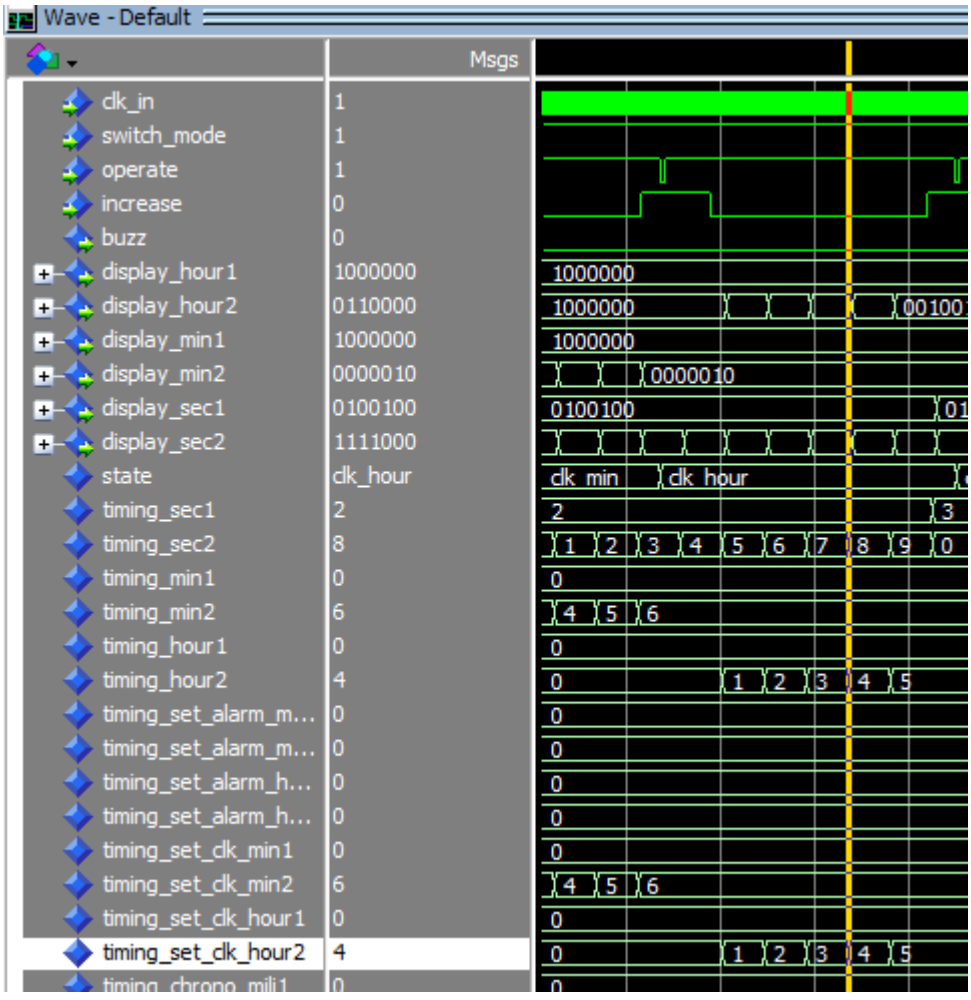


This screenshot shows the Clock status. The clock is running normally. Our clock cycle is based on clk_in, in the code by

changing the count to specify how many clk_in in one second.

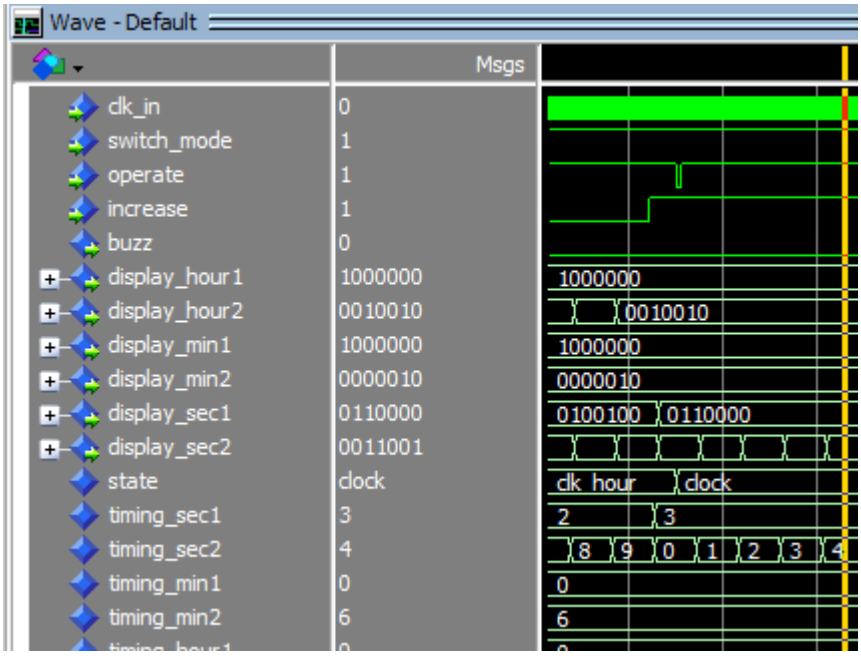


In the Clk state, if operate = 0, it will enter the clk_min state. In this state, we can change the "min" displayed by the clock. If increase = 0, it will increase min to min + 1. As can be seen in the screenshot, after entering the clk_min state, the value of min can be successfully changed after increase = 0. Timing_set_clk_min represents the set time.

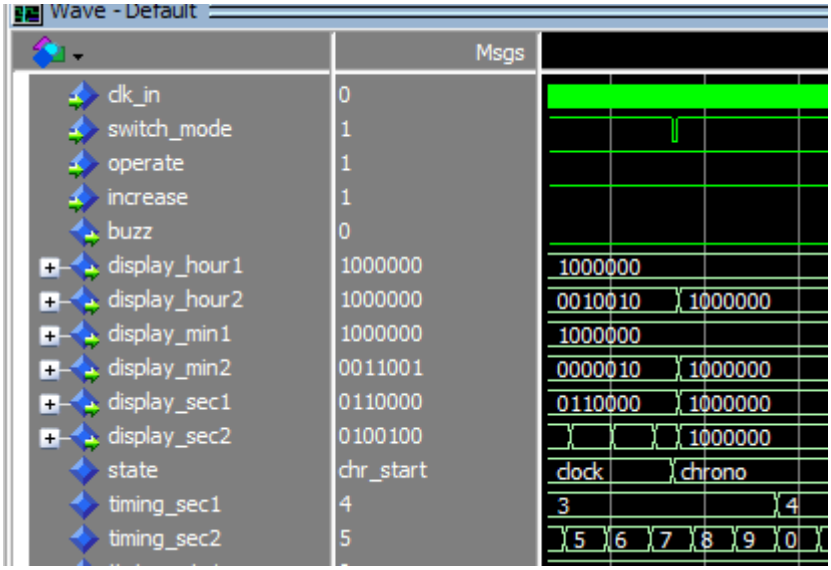


In the clk_min state, if operate = 0, it will enter the clk_hour state. This state is similar to clk_min and is used to set the clock

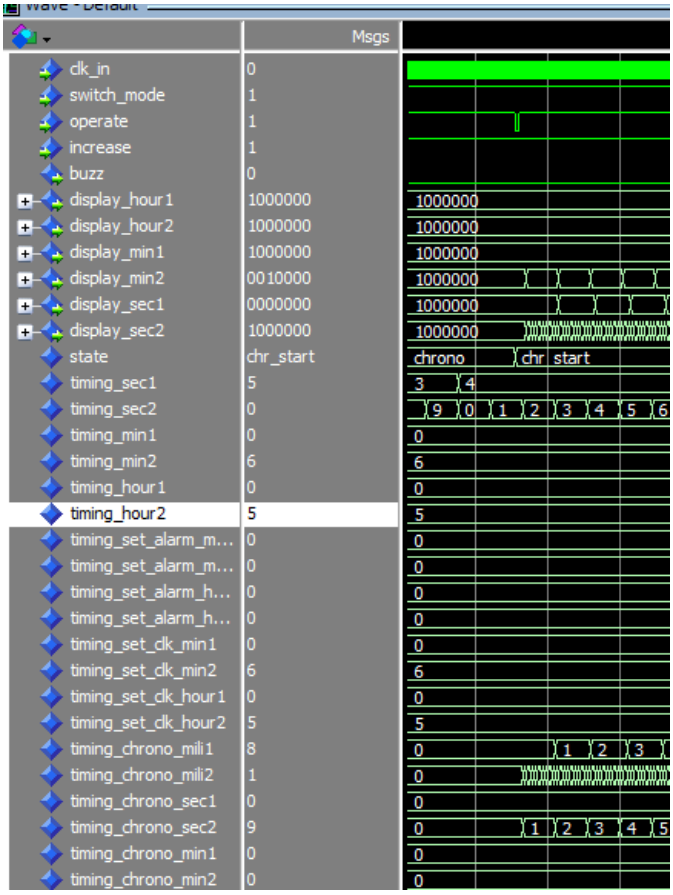
"hour". It can be seen in the screenshot that the clock "hour" is successfully set. Timing_set_clk_hour is used to set the hour.



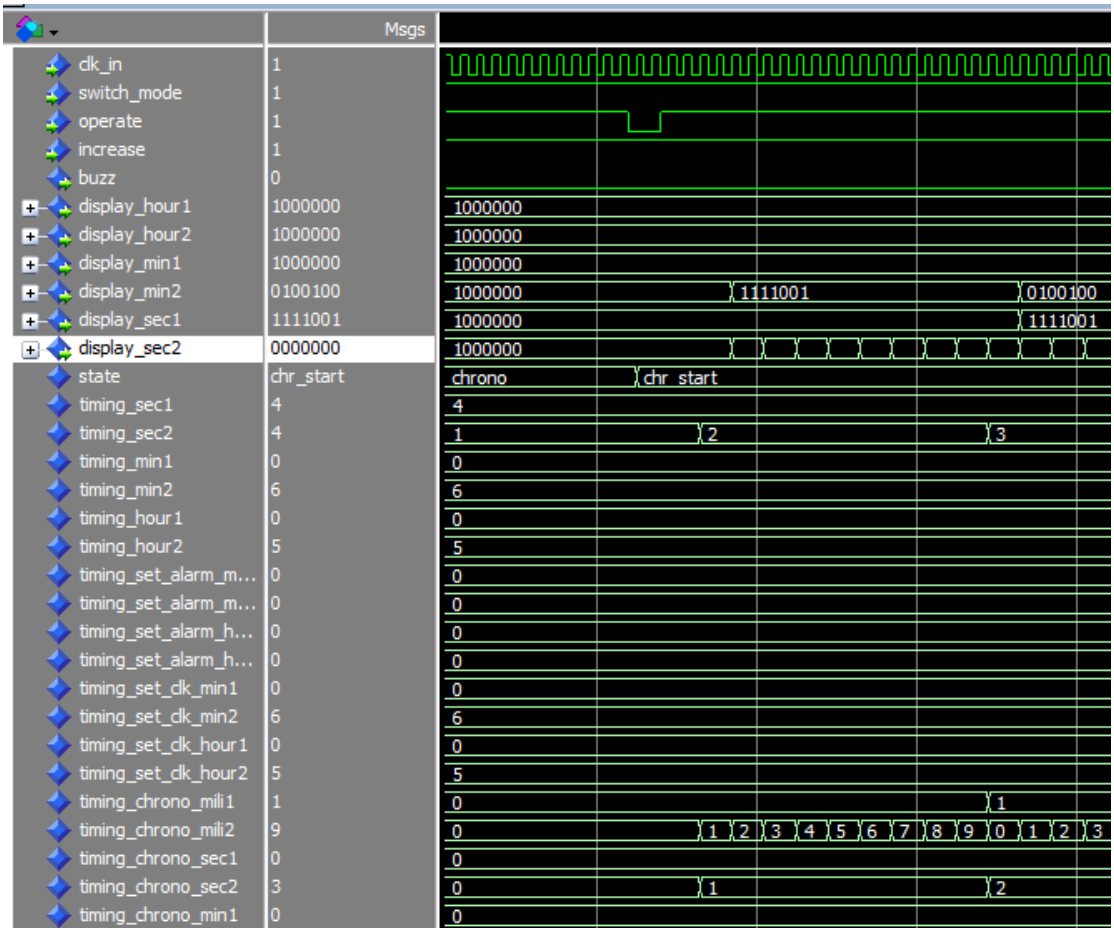
In the clk_hour state, if operate = 0 will return to the clock state. The screenshot shows a successful return to the clock state.



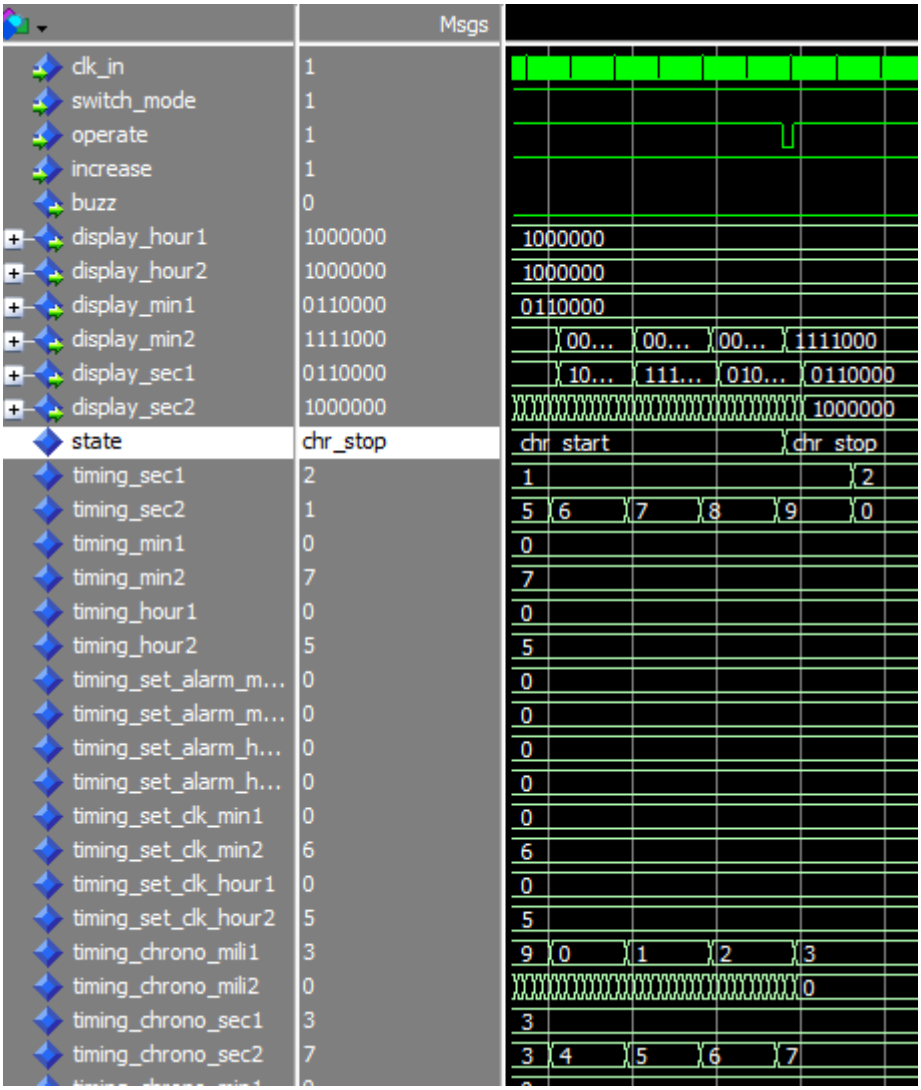
In the clock state, if switch_mode = 0, it will enter the chrono state. This state is related to the stopwatch function. In this screenshot, the chrono state of successful transition is shown.



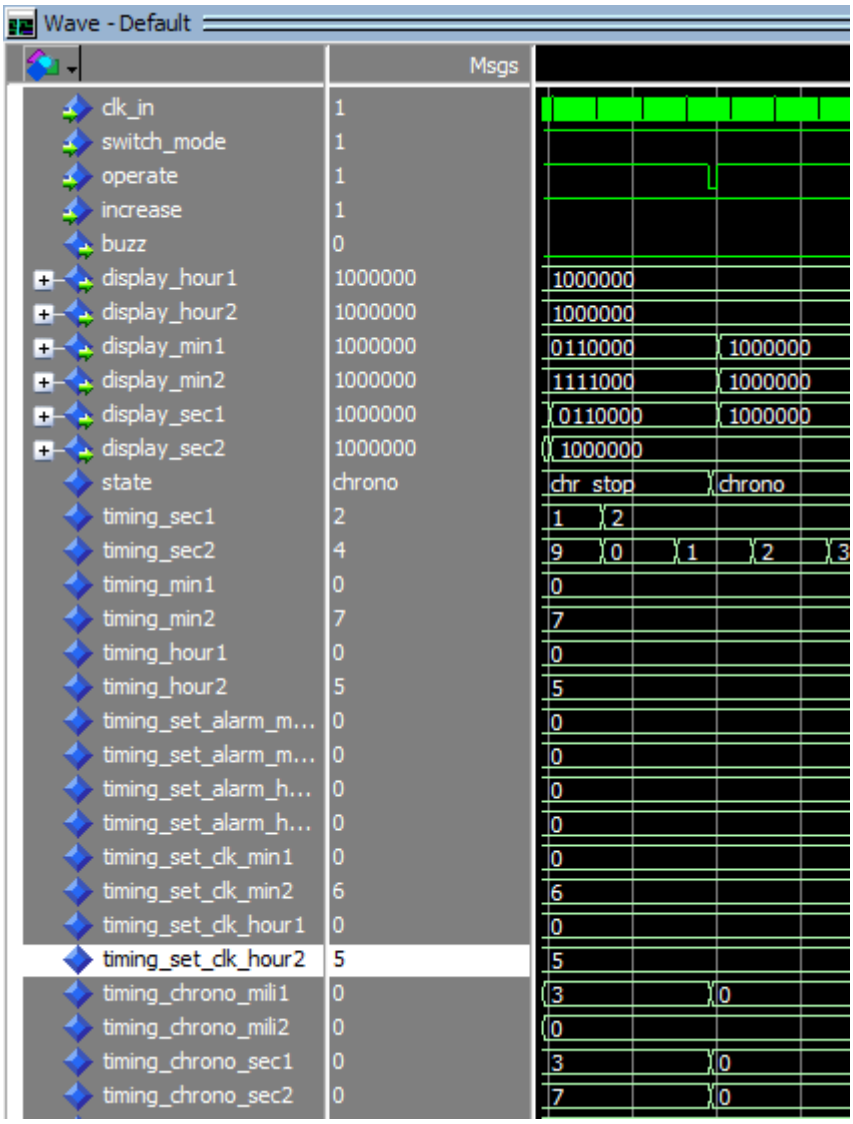
In the chrono state, if operate = 0, it will enter the chr_start state. The stopwatch will start timing in this state. The screenshot shows a successful transition from the chrono state to the chr_start state.



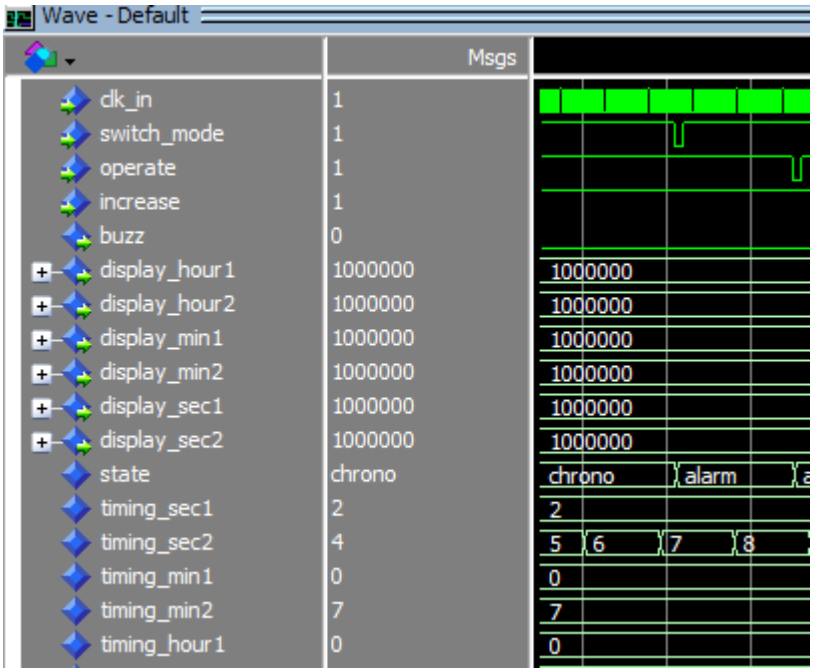
In the chr_start state, the stopwatch starts timing. The successful timing is shown in the screenshot. The parameters related to timing_chrono show the stopwatch information.



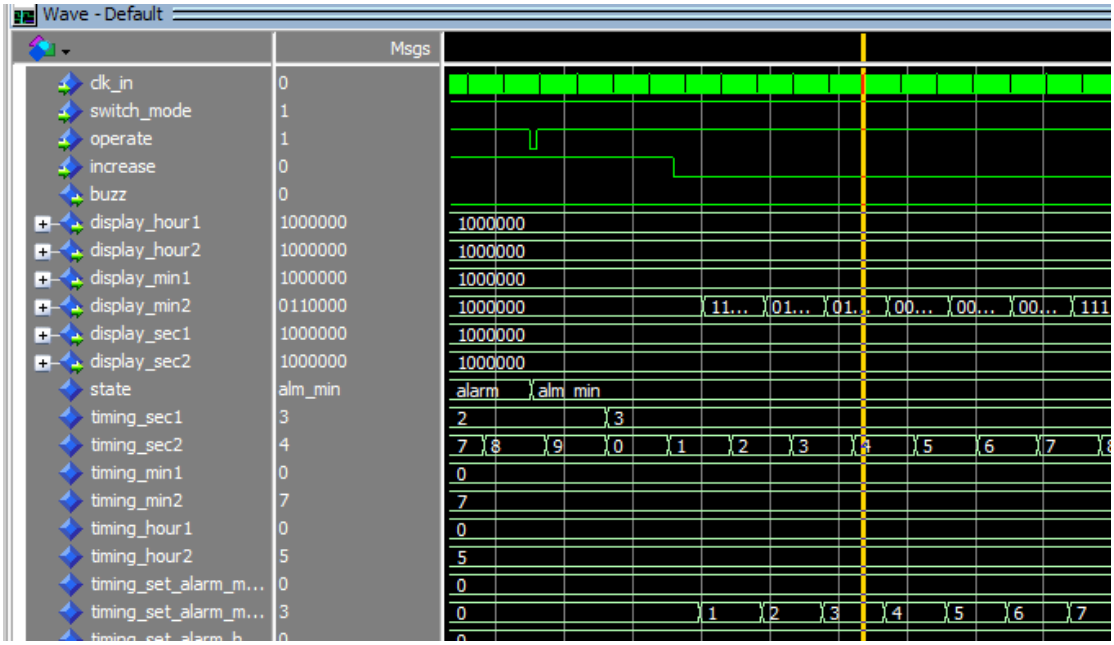
In the chr_start state, it will enter the chr_stop state when operate = 0. In this state, it will stop timing. The stopwatch also displays the previously recorded time. This screenshot shows the successful recording of the stopwatch time and stop timing after entering the chr_stop state.



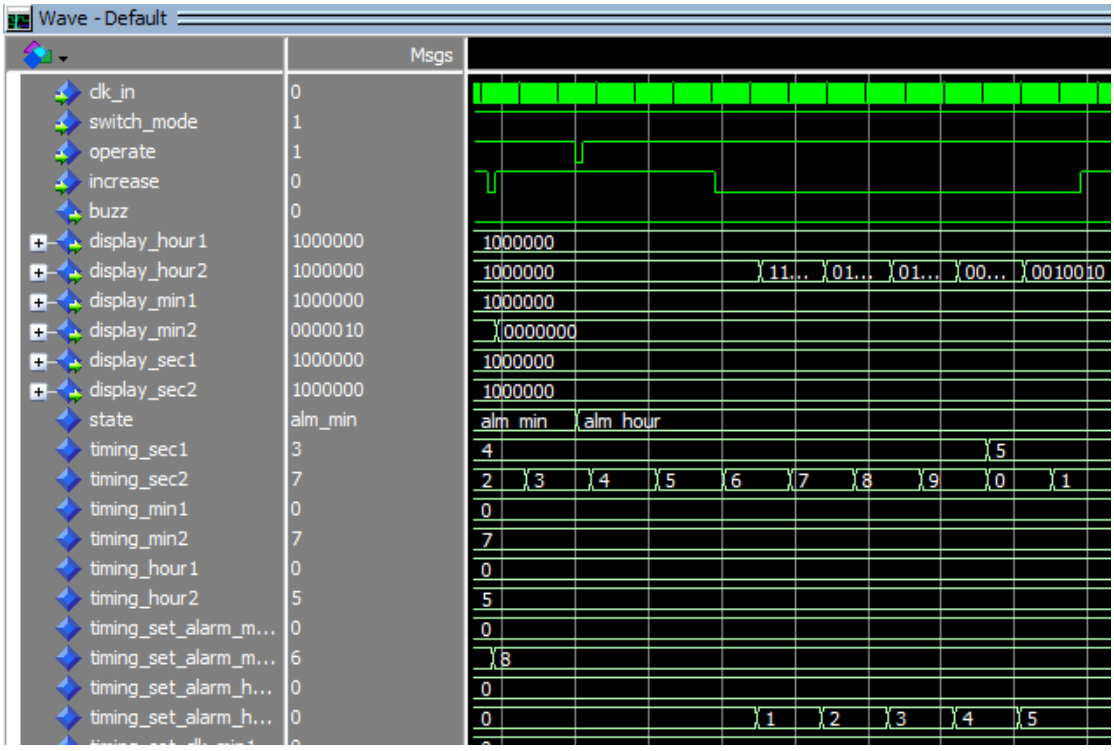
In the chr_stop state, if operate = 0, it will return to the chrono state, and the chrono_timing will be cleared to 0. This screenshot shows the successful return to the chrono state.



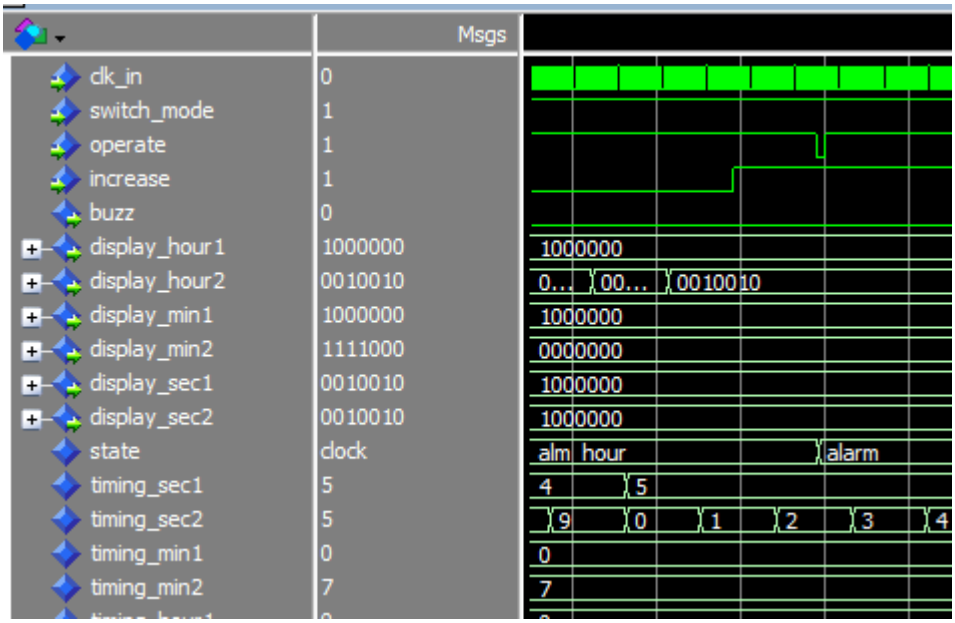
In the chrono state, if switch_mode = 0, it will switch to the alarm state. This screenshot shows the successful switch to alarm mode



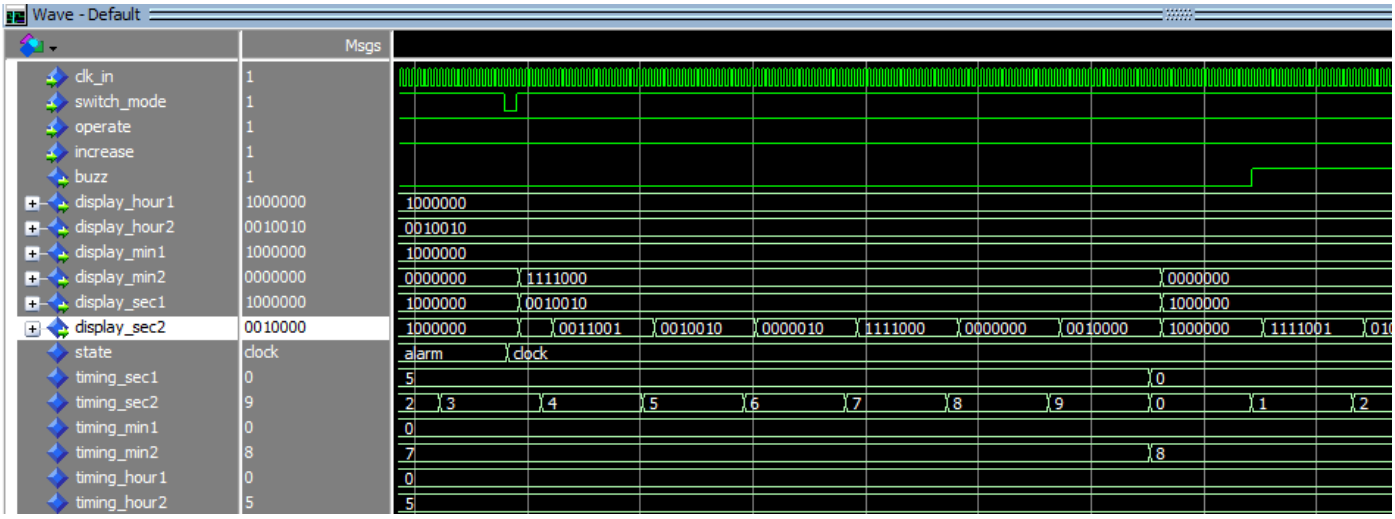
In the alarm state, if operate = 0, it will enter the alm_min state. In this state, if increment = 0, the alarm "min" will be increased. The screenshot shows the successful increase of alarm minutes.



In the alm_min state, if operate-0 will enter the alm_hour state. In this state, if increment = 0, the "hour" of the alarm will be increased. The screenshot shows the successful increase of alarm hours. Here we can see the alarm time is set as 05:08.



In the alm_hour state, if operate = 0 will return to the alarm state. The screenshot shows the successful return to the alarm state.



In the alarm state, if switch_mode = 0 will return to the clock state. The screenshot shows the successful return to the clock state. At the same time, you can also see in this screenshot that buzz = 1 will start ringing when the clock time is the same as the alarm time which is 05:08.

6. Conclusion

The digital clock we designed in the project has three functions: clock, alarm clock, and stopwatch. The clock can be displayed on the digital tube, showing the corresponding hours, minutes, and seconds. Modifying the corresponding count in the code can change the frequency to achieve a real-world clock. The alarm clock function can set a time when the clock reaches this event; it will start ringing, corresponding to buzz = 1 in the code. The stopwatch function can accurately record the time after the start of the timing. Since the Quartus or ModelSim we use are tough to simulate a considerable time, we just set the frequency of milliseconds in the code to be several times of seconds, rather than 1000 times accurate. Overall, we have achieved what we wanted to accomplish in the expected goal.