UPPSALA
UNIVERSITET

Report for 1FA326

Project: Digital Alarm Clock

Kleivi Tsaousi

April 17, 2019

Abstract

The project task is to design a 24h alarm clock the Altera DE2 FPGA board. The user has the ability to reconfigure the time of the clock and also the time of the alarm to turn on. The time and alarm is displayed on the 7-Segments and also on the LCD screen of the Altera board. The final design contains all the entities which combined, implement the desired functionality of the project.

Contents

1 Introduction **4**

2 Project Description **5**

2.1 Tools and Software 5

2.2 Design 5

3 Theory **6**

3.1 Clock Devider 6

3.2 Control 6

3.3 Clock Counter 6

3.4 Alarm Clock 6

3.5 Comparator 7

3.6 MUX (for 7 Segment) 7

3.7 7-Segment 7

3.8 LCD 7

4 Implementation **8**

4.1 Clock Devider 8

4.2 Control 8

4.3 Clock 9

4.4 Alarm 10
4.5 Comparator 11
4.6 MUX and 7-Segments 12
4.7 Complete project of 24h digital alarm clock 13

5 Test & Results **14**

5.1 Signal Tap 14

6 Conclusion & Evaluation **16**

7 Appendix **17**

7.1 Clock Devider 17
7.2 Control 18
7.3 Clock Counter 19
7.4 Alarm Counter 21
7.5 Comparator 23
7.6 MUX 24

8 References **27**

1 Introduction

This project gives the possibility to set a 24 hours digital clock and an alarm clock, so that when the time comes, the alarm indication will turn on an LED. This project also gives a good and valuable knowledge about multiple entities and how to port map them together. Also how internal clocks work, together with switches, buttons, 7 Segments, LEDs and LCDs.

The final design consists of different entities which share same inputs some times. This is done in order for the project to be more manageable but also for the programmer not be forced to use a very big piece of code.

2 Project Description

2.1 Tools and Software

The project task is to design a 24 hours alarm clock on the Altera DE-2 board. The user can set the minutes and hours of the clock and also the time of the alarm. The time and alarm will be displayed on the 7-Segments and LCD of the FPGA board.

The code implementation was done in Quartus v.13 using VHDL programming language.

2.2 Design

The design is using smaller components, organized as entities in order for the system to be more manageable. These units are:

- Clock Devider
- Control
- Clock Counter
- Alarm Counter
- Comparator
- MUX (for 7-Segment)
- 7-Segment
- LCD Control

The units perform simple tasks that together compose the final project of the digital clock and the alarm application. The time and alarm is displayed on the 7 segment by choosing the switch state. Also time and alarm is displayed on the LCD screen.

3 Theory

3.1 Clock Devider

The clock devider takes as an input the internal 50 MHz clock of the FPGA board. As output gives out the 1 Hz counter which drives the clock counter for 1 second. Also gives out the 10 Hz counter which drives the clock counter which increments the values of minutes and hours according to the user. The last 400 Hz counter drives the counter which is used for the LCD display.

3.2 Control

The control unit takes as inputs the 1 Hz and 100 Hz clock counter from the previous entity. It also takes the ”set”, ”mode” and ”reset” parameters from the user. The ”reset” switch erases all the registers both clock and alarm. The ”mode” switch controls either the clock or the alarm mode. The ”set” switch, when enabled, allows the user to change the time of the clock and the alarm. Finally it gets the indication if the alarm should turn on. As output it gives the 1 Hz and 100 Hz counters. It also gives the ”set”, ”mode” and ”set” switches. Also drives the LED output if it is time to turn on.

3.3 Clock Counter

The clock counter unit takes as input the 1 Hz and 100 Hz clock counter from the previous entity. Also takes the button to increment the minutes and the button that increments the hours. The ”set”, ”mode” and ”reset” switches also come as inputs. If the ”reset” switch is enabled, the registers for the clock will be erased and become zero. For every rising edge of the 1 Hz clock counter, and the ”set” switch is not enabled, the clock will increment by 1 second. Otherwise, if the ”set” switch is enabled, the clock freezes and the user can change the time with the buttons for the minutes and time accordingly. The outputs of the clock entity are the six vectors which store the values of 2 digits of the hour, 2 digits of the minutes and 2 digits for the seconds.

3.4 Alarm Clock

The alarm clock entity takes as input the 1 Hz and 100 Hz clock counter from the previous entity. Also uses the same buttons as as the clock counter in order for the user to change the time of the alarm clock to turn on the indication. The ”set”, ”mode” and ”reset” switches come also as inputs. If the ”mode” switch is set for the alarm and the ”set” switch is enabled, the user can change the time of the alarm. If the ”set” switch is not enabled, then he can not change the time

with the given buttons. As output the entity gives out the vectors which store the values of the 2 digits of the hours and 2 digits of the minutes of the alarm.

3.5 Comparator

The comparator entity takes as input the 6 vectors from the clock entity and the 4 vectors from the alarm entity. Then it compares them and in case the 2 digits of the hour from the clock are equal with their according to the alarm, and also the minute digits are the same, it gives out an indication which goes to the control entity in order to turn on the alarm.

3.6 MUX (for 7 Segment)

The MUX entity takes as input the 6 vectors of the clock entity, together with the 4 vectors of the alarm entity and the ”mode” switch. If the ”mode” is selected at 0 state, it displays the the clock on the on the 7-Segment displays. On the other hand, if it is at 1 state, it displays the alarm on the 7-Segment displays. Since we don't care care about the seconds on the alarm, the values remain the same as in the clock mode. The vectors are modified as 6 downto 0 vectors in order to be able to be displayed on the 7-Segments. These vectors are declares as output in order to drive the final indication.

3.7 7-Segment

Each 7-Segment consists of a 6 downto 0 vector. Each digit from the clock counter or the alarm counter requires a 7-Segment. All of them together consist of the total clock time and alarm indication. When the user switches through the modes, the alarm digit hours are displayed in the place where also the clock digit hours are displayed. This goes also to the minutes. However, the alarm does not have anything to do with the seconds.

3.8 LCD

The lcd takes as input the 400 Hz clock counter which comes from the clock devider. Also takes the vectors from the lcd but first they have to be converted to hexadecimal numbers. The conversion is based on the ASCII table in order to represent each number or character. On the First line is written the clock counter and on the second the alarm counter.

4 Implementation

4.1 Clock Devider

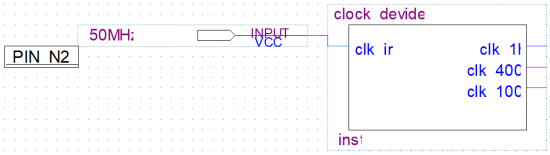

Figure 1: Entity of clock devider. One input and three outputs.

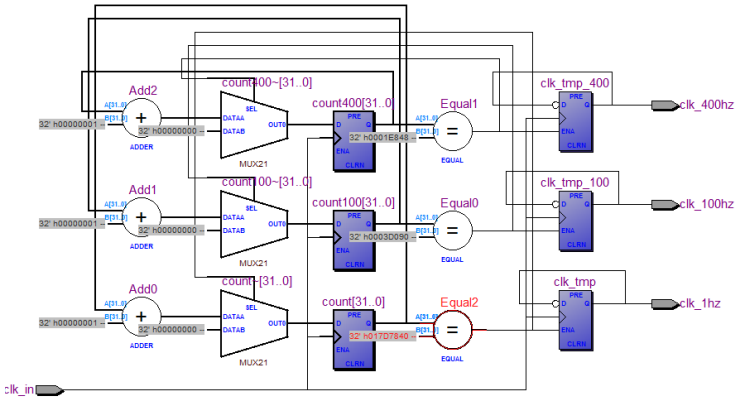

Figure 2: RTL view of clock devider.

4.2 Control

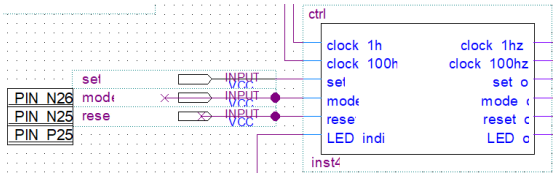

Figure 3: Entity of clock control. Six input and six outputs.

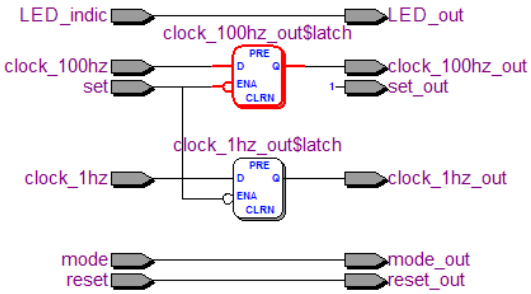

Figure 4: RTL view of control

4.3 Clock

Figure 5: Entity of clock . Seven input and six outputs.

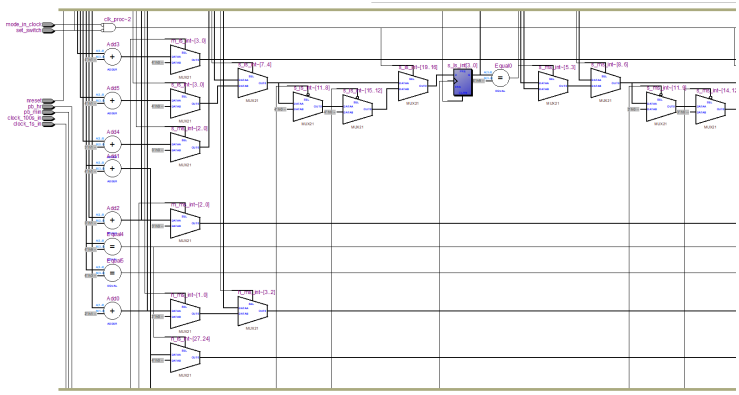

Figure 6: RTL 1 view of clock devider

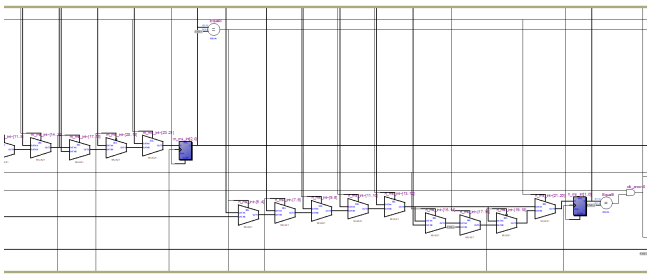

Figure 7: RTL 2 view of clock devider

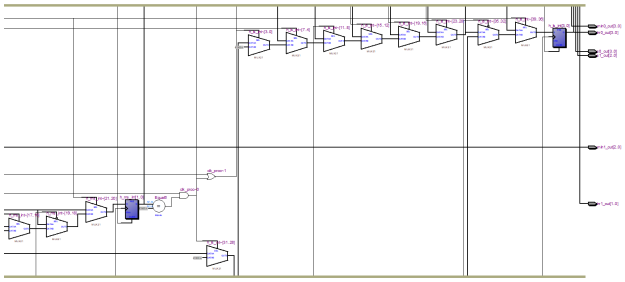

Figure 8: RTL 3 view of clock devider

4.4 Alarm

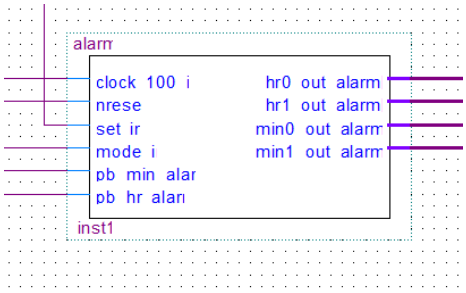

Figure 9: Entity of alarm. Six inputs and four outputs

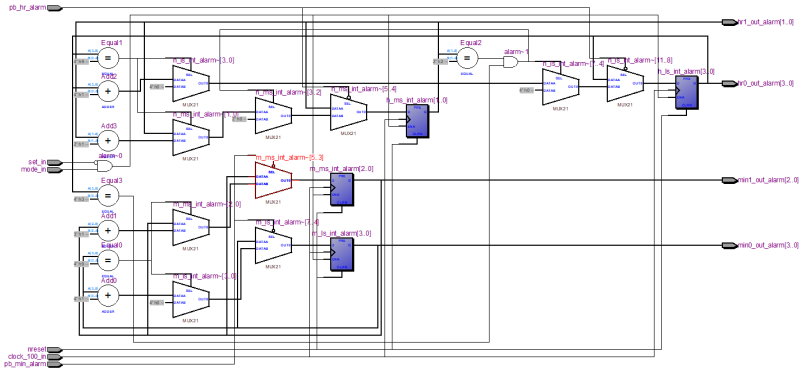

Figure 10: RTL view of alarm

4.5 Comparator

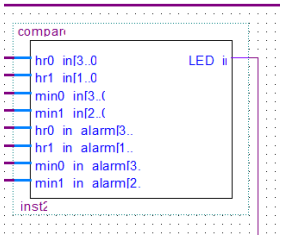

Figure 11: Entiry of comparator. Eight inputs and one

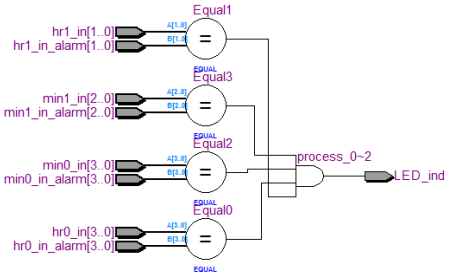

Figure 12: RTL view of comparator

4.6 MUX and 7-Segments

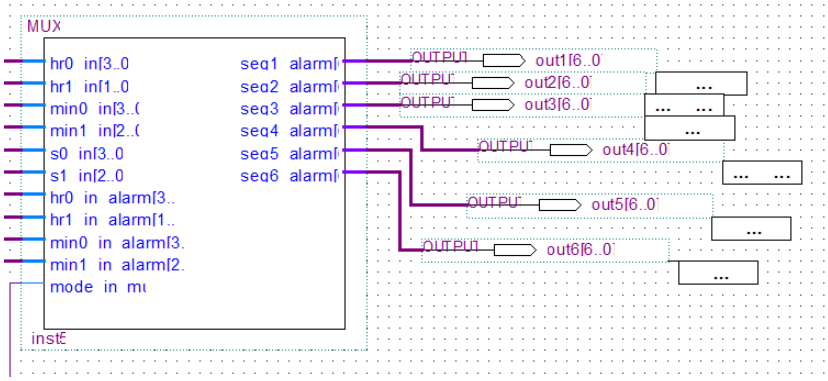

Figure 13: Entity of MUX and 7-Segments. Eleven inputs and six outputs

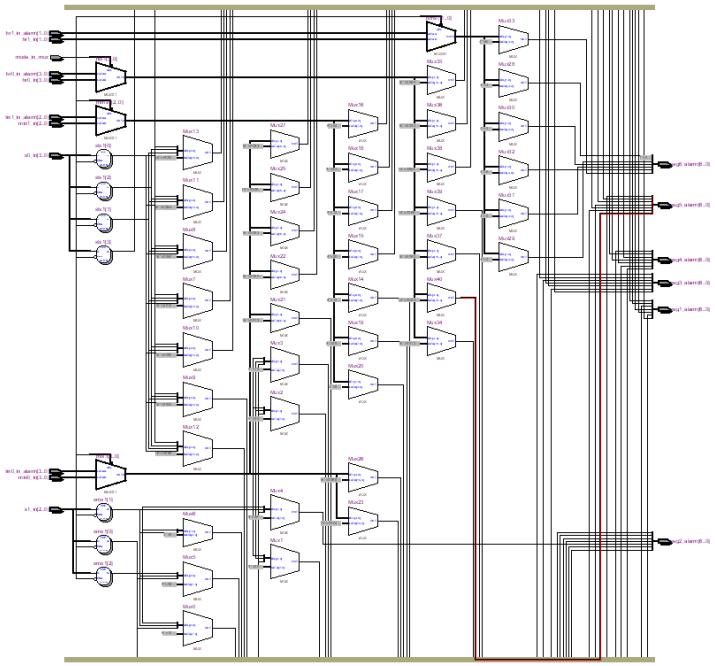

Figure 14: RTL view of MUX and 7-Segments

4.7 Complete project of 24h digital alarm clock

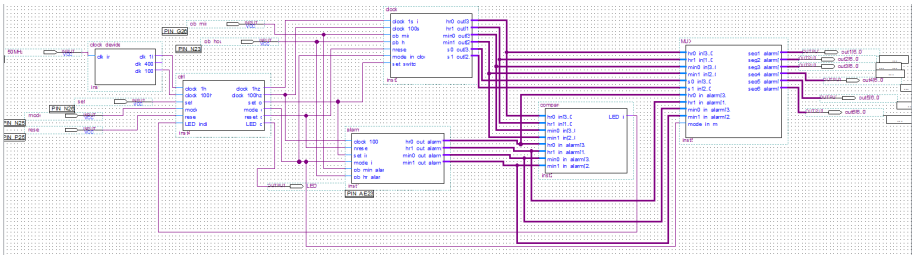

Figure 15: Complete entity of the project

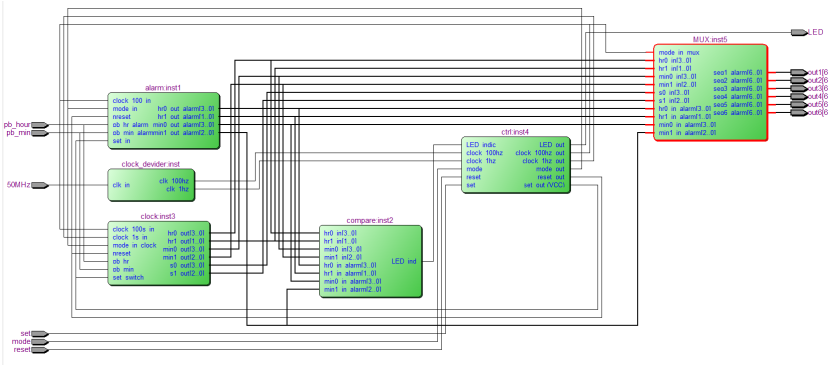

Figure 16: RTL view of project

5 Test & Results

5.1 Signal Tap

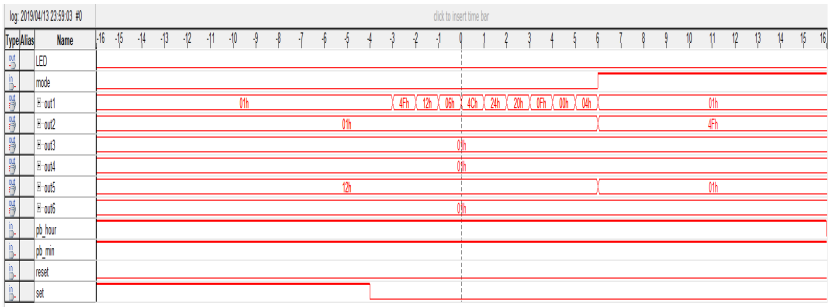

Figure 17: Signal Tap of project (1 pt)

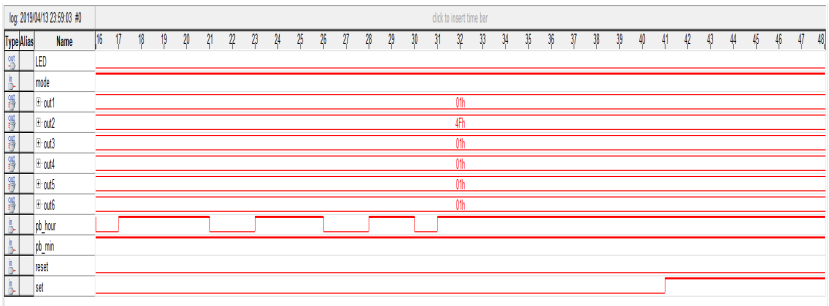

Figure 18: Signal Tap of project (2pt)

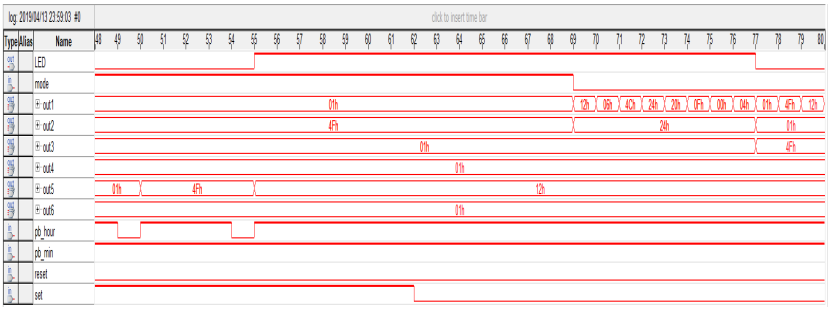

Figure 19: Signal Tap of project (3pt)

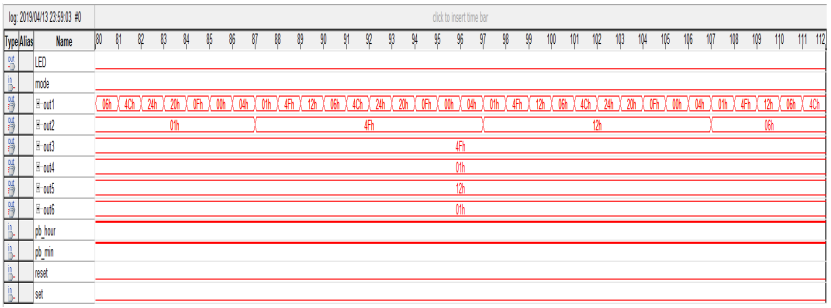

Figure 20: Signal Tap of project (4pt)

One of the requirements for this project is to change to the time every one seconds as it happens in every normal clock. Also every time the second counter reaches at 60, the the minutes counter should increment. The same applies to the minute counter as they reach the value 60, it should increment the hour counter. When the hour counter reaches 24, it should erase all registers, as it happens at exactly midnight. Results of the execution show that all clock counters work as they should be. Also when the ”reset” switch is on, all registers become zero. When the ”switch” mode is on, the hour and minute button increment the minutes and clock counters. Finally when the clock time matches the alarm time, the LED indications turns on.

6 Conclusion & Evaluation

As seen from the Signal Tap simulation and the Hardware Testing, the clock counters work as they were expected to be. Also by switching ”mode” the user is able to change between clock and alarm selection. When the ”set” switch is enabled, the user is able to change to adjust the time and alarm according to his preferences. Also the LED indication turn on when the alarm matches the time of the clock. When the ”reset” switch is enabled, all the registers from clock and alarm, go back to zero. All the displays of the digits are done through the 7-Segment displays. However the implementation of the LCD was not able to be completed through the complication of the module. The buttons that change the clock counter and alarm counter run with a frequency of 100 Hz that comes from the clock devider. This is used in order for the counters to run on a faster speed.

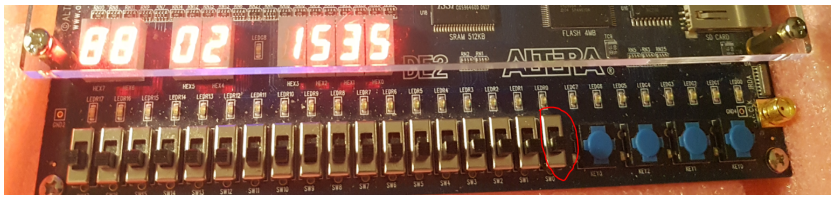

Figure 21: Clock mode on the board

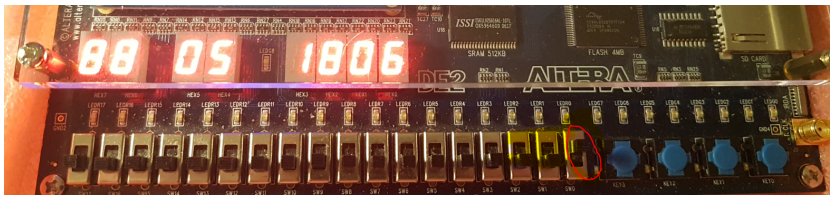

Figure 22: Alarm mode on the board

Figure 23: Alarm indication on the board

7 Appendix

7.1 Clock Devider

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned;

entity clock_devider is
port(
        clk_in: in std_logic;
        clk_1hz: out std_logic; --for main clock
        clk_400hz: out std_logic; --for LCD
        clk_100hz: out std_logic --for set alarm
        );
end clock_devider;

architecture bhv of clock_devider is
    signal clk_tmp, clk_tmp_100 , clk_tmp_400: std_logic :='0';
    signal count, count100, count400 : integer :=0;

    begin
     process (clk_in, clk_tmp, clk_tmp_100, clk_tmp_400) begin
        if (rising_edge(clk_in)) then
            count <= count +1;
            count100<=count100+1;
            count400<=count400+1;
            if (count100 = 2500000) then
                clk_tmp_100 <= not clk_tmp_100;
                count100<=0;
            end if;
            if (count400 = 125000) then
                clk_tmp_400<= not clk_tmp_400;
                count400<=0;
            end if;
            if (count = 2500000) then
                clk_tmp <= not clk_tmp;
                count <= 0;
            end if;
        end if;
        clk_1hz <= clk_tmp;
        clk_100hz <= clk_tmp_100;
        clk_400hz <=clk_tmp_400;
    end process;
end architecture bhv;
```

7.2 Control

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity ctrl is
8  port(
9          clock_1hz: in std_logic;
10         clock_1hz_out: out std_logic;
11         clock_100hz: in std_logic;
12         clock_100hz_out: out std_logic;
13         set: in std_logic;
14         set_out : out std_logic;
15         mode: in std_logic;
16         mode_out : out std_logic;
17         reset: in std_logic;
18         reset_out: out std_logic;
19         LED_indic: in std_logic;
20         LED_out: out std_logic
21
22         );
23 end entity ctrl;
24
25 architecture behaviour of ctrl is
26     signal set_tmp, mode_tmp, reset_tmp: std_logic:='0';
27
28 begin
29         clock_100hz_out<=clock_100hz;
30             clock_1hz_out<=clock_1hz;
31
32     process(set, set_tmp) begin --set button to change time and alarm
33         if (set='1') then
34             set_tmp<='1';
35         else
36             set_tmp<='0';
37         end if;
38         set_out<=set_tmp;
39     end process;
40
41     process(mode, mode_tmp) begin   --switch modes clock/alarm
42         if (mode='1') then
43             mode_tmp<='1';
44         else
45             mode_tmp<='0';
46         end if;
47         mode_out<=mode_tmp;
48     end process;
49
50     process(reset, reset_tmp) begin --reset registers
51         if (reset='1') then
52             reset_tmp<='1';
53         else
54             reset_tmp<='0';
55         end if;
56         reset_out<=reset_tmp;
57     end process;
58
59     process(LED_indic) begin    --LED indication when the alarm rings
60         if (LED_indic='1') then
61             LED_out<='1';
62         else
63             LED_out<='0';
64         end if;
65     end process;
66
67 end architecture behaviour;
```

7.3 Clock Counter

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity clock is
8  port(
9          clock_1s_in: in std_logic;
10         clock_100s_in: in std_logic;
11         pb_min: in std_logic;
12         pb_hr: in std_logic;
13         nreset: in std_logic;
14         mode_in_clock: in std_logic;
15         set_switch:in std_logic;
16         hr0_out: out std_LOGIC_VECTOR(3 downto 0);
17         hr1_out: out std_LOGIC_VECTOR(1 downto 0);
18         min0_out: out std_LOGIC_VECTOR(3 downto 0);
19         min1_out: out std_LOGIC_VECTOR(2 downto 0);
20         s0_out: out std_LOGIC_VECTOR(3 downto 0);
21         s1_out: out std_LOGIC_VECTOR(2 downto 0)
22
23
24         );
25 end entity clock;
26
27 architecture bhv_clk of clock is
28
29      signal h_ms_int : integer range 0 to 2;
30     signal h_ls_int : integer range 0 to 9;
31     signal m_ms_int : integer range 0 to 5;
32     signal m_ls_int : integer range 0 to 9;
33     signal s_ms_int : integer range 0 to 5;
34     signal s_ls_int : integer range 0 to 9;
35
36     signal sls :   STD_LOGIC_VECTOR(3 DOWNTO 0);
37     signal sms :   STD_LOGIC_VECTOR(2 DOWNTO 0);
38     signal mls :   STD_LOGIC_VECTOR(3 DOWNTO 0);
39     signal mms :   STD_LOGIC_VECTOR(2 DOWNTO 0);
40     signal hls :   STD_LOGIC_VECTOR(3 DOWNTO 0);
41     signal hms :   STD_LOGIC_VECTOR(1 DOWNTO 0);
42
43     begin
44
45     -- Process that handles the project logic
46         clk_proc : process(clock_1s_in, nreset, h_ms_int, h_ls_int, m_ms_int,
             m_ls_int, s_ms_int, s_ls_int, mode_in_clock, set_switch)
47             begin
48             if nreset = '1' then --if "reset" pressed
49                 h_ms_int <= 0;
50                 h_ls_int <= 0;
51                 m_ms_int <= 0;
52                 m_ls_int <= 0;
53                 s_ms_int <= 0;
54                 s_ls_int <= 0;
55             elsif (clock_1s_in'event and clock_1s_in='1') then
56                 if (set_switch='0') then --if switch mode is disabled
57                 if s_ls_int = 9 then
58                     if s_ms_int = 5 then
59                         if (m_ls_int = 9) then
60                             if (m_ms_int = 5 ) then
61                                 if (h_ls_int = 9 or (h_ls_int = 3 and
                                        h_ms_int = 2) ) then
62                                 h_ls_int <= 0;
63                                     if ((h_ls_int=3 and h_ms_int=2)) then
64                                         h_ms_int <= 0;
65                                         h_ls_int <= 0;
66                                         m_ms_int <= 0;
67                                         m_ls_int <= 0;
68                                         s_ms_int <= 0;
69                                         s_ls_int <= 0;
```

```
70                                          else
71                                             h_ms_int <= h_ms_int + 1;
72                                          end if;
73                                      h_ls_int <= 0;
74                                  else
75                                      h_ls_int <= h_ls_int + 1;
76                                  end if;
77                                  m_ms_int <= 0;
78                              else
79                                  m_ms_int <= m_ms_int + 1;
80                              end if;
81                              m_ls_int <= 0;
82                          else
83                              m_ls_int <= m_ls_int + 1;
84                          end if;
85                          s_ms_int <= 0;
86                      else
87                           s_ms_int <= s_ms_int + 1;
88                      end if;
89                      s_ls_int <= 0;
90                  else
91                       s_ls_int <= s_ls_int + 1;
92                          end if;
93                  end if;
94
95
96          if ( mode_in_clock='0' and set_switch='1') then --if "switch" mide
                enabled
97              if pb_min='0' then --press button for minutes
98                  m_ls_int<=m_ls_int + 1;
99                  s_ls_int<=0;
100                 s_ms_int<=0;
101                 if m_ls_int=9 then
102                     m_ls_int<=0;
103                     m_ms_int<=m_ms_int+1;
104                 end if;
105             end if;
106             if pb_hr='0' then --press button for hours
107                 h_ls_int<=h_ls_int+1;
108                 s_ls_int<=0;
109                 s_ms_int<=0;
110                      if h_ls_int=9 then
111                         h_ls_int<=0;
112                   h_ms_int<=h_ms_int+1;
113                     end if;
114                     if (h_ms_int=2 and h_ls_int=3) then
115                        h_ms_int<=0;
116                        h_ls_int<=0;
117                     end if;
118                  end if;
119
120              end if;
121         end if;
122
123
124
125
126
127 end process clk_proc;
128
129     hr0_out <= std_LOGIC_VECTOR(to_unsigned(h_ls_int, hls'length));
130     hr1_out <= std_LOGIC_VECTOR(to_unsigned(h_ms_int, hms'length));
131     min0_out <= std_LOGIC_VECTOR(to_unsigned(m_ls_int, mls'length));
132     min1_out <= std_LOGIC_VECTOR(to_unsigned(m_ms_int, mms'length));
133     s0_out <= std_LOGIC_VECTOR(to_unsigned(s_ls_int,sls'length));
134     s1_out <= std_LOGIC_VECTOR(to_unsigned(s_ms_int,sms'length));
135
136 end architecture;
```

7.4 Alarm Counter

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity alarm is
8  port(
9          clock_100_in : in std_logic;
10         nreset: in std_logic;
11         set_in: in std_logic;
12         mode_in: in std_logic;
13         pb_min_alarm: in std_logic;
14         pb_hr_alarm: in std_logic;
15         hr0_out_alarm: out std_logic_vector(3 downto 0);
16         hr1_out_alarm: out std_LOGIC_VECTOR(1 downto 0);
17         min0_out_alarm: out std_LOGIC_VECTOR(3 downto 0);
18         min1_out_alarm: out std_LOGIC_VECTOR(2 downto 0)
19
20         );
21 end entity alarm;
22
23 architecture bhv_alarm of alarm is
24
25      signal h_ms_int_alarm : integer range 0 to 2;
26     signal h_ls_int_alarm : integer range 0 to 9;
27     signal m_ms_int_alarm : integer range 0 to 5;
28     signal m_ls_int_alarm : integer range 0 to 9;
29
30
31     signal mls_alarm :   STD_LOGIC_VECTOR(3 DOWNTO 0);
32     signal mms_alarm :   STD_LOGIC_VECTOR(2 DOWNTO 0);
33     signal hls_alarm :   STD_LOGIC_VECTOR(3 DOWNTO 0);
34     signal hms_alarm :   STD_LOGIC_VECTOR(1 DOWNTO 0);
35
36     begin
37
38     process(set_in, clock_100_in, nreset, h_ms_int_alarm, h_ls_int_alarm,
           m_ms_int_alarm, m_ls_int_alarm, mode_in, pb_min_alarm, pb_hr_alarm)
39     begin
40
41         if (nreset='1') then --if "reset" is pressed
42             h_ms_int_alarm <= 0;
43             h_ls_int_alarm <= 0;
44             m_ms_int_alarm <= 0;
45             m_ls_int_alarm <= 0;
46
47         elsif (mode_in='1' and set_in='1' ) then --if "mode" is alarm and "
               set" enabl
48         if (clock_100_in'event and clock_100_in='1') then
49             if pb_min_alarm='0' then --if button for minutes pressed
50                 m_ls_int_alarm<=m_ls_int_alarm + 1;
51                 if m_ls_int_alarm=9 then
52                     m_ls_int_alarm<=0;
53                     m_ms_int_alarm<=m_ms_int_alarm+1;
54                 end if;
55             end if;
56             if pb_hr_alarm='0' then --if button for hours pressed
57                 h_ls_int_alarm<=h_ls_int_alarm+1;
58                      if h_ls_int_alarm=9 then
59                         h_ls_int_alarm<=0;
60                   h_ms_int_alarm<=h_ms_int_alarm+1;
61                      end if;
62                      if (h_ms_int_alarm=2 and h_ls_int_alarm=3) then
63                         h_ms_int_alarm<=0;
64                         h_ls_int_alarm<=0;
65                      end if;
66             end if;
67                 end if;
68         end if;
69
```

```
70
71
72          hms_alarm <= std_logic_vector(to_unsigned(h_ms_int_alarm, hms_alarm'
                length));
73          hls_alarm <= std_logic_vector(to_unsigned(h_ls_int_alarm, hls_alarm'
                length));
74          mms_alarm <= std_logic_vector(to_unsigned(m_ms_int_alarm, mms_alarm'
                length));
75          mls_alarm <= std_logic_vector(to_unsigned(m_ls_int_alarm, mls_alarm'
                length));
76
77          hr0_out_alarm <= std_LOGIC_VECTOR(to_unsigned(h_ls_int_alarm,
                hls_alarm'length));
78          hr1_out_alarm <= std_LOGIC_VECTOR(to_unsigned(h_ms_int_alarm,
                hms_alarm'length));
79          min0_out_alarm <= std_LOGIC_VECTOR(to_unsigned(m_ls_int_alarm,
                mls_alarm'length));
80          min1_out_alarm <= std_LOGIC_VECTOR(to_unsigned(m_ms_int_alarm,
                mms_alarm'length));
81          end process alarm;
82
83  end architecture bhv_alarm;
```

7.5 Comparator

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity compare is
8  port( hr0_in: in std_logic_vector (3 downto 0);
9          hr1_in: in std_logic_vector (1 downto 0);
10         min0_in: in std_logic_vector (3 downto 0);
11         min1_in: in std_logic_vector (2 downto 0);
12
13         hr0_in_alarm: in std_logic_vector (3 downto 0);
14         hr1_in_alarm: in std_logic_vector (1 downto 0);
15         min0_in_alarm: in std_logic_vector (3 downto 0);
16         min1_in_alarm: in std_logic_vector (2 downto 0);
17
18         LED_ind: out std_logic
19
20         );
21 end entity compare;
22
23 architecture cmp_bhv of compare is
24     signal led_tmp: std_logic:='0';
25
26     begin
27
28     process(hr0_in, hr0_in_alarm, hr1_in, hr1_in_alarm, min0_in,
           min0_in_alarm, min1_in, min1_in_alarm, led_tmp)
29     begin
30         if (hr0_in=hr0_in_alarm and hr1_in=hr1_in_alarm and min0_in=
               min0_in_alarm and min1_in=min1_in_alarm) then --if the hour and
31                                                  --  minutes of the hour
32                 led_tmp<='1';    --are the same, send the indication
33         else
34             led_tmp<='0';
35         end if;
36         LED_ind<=led_tmp;
37     end process;
38
39
40 end architecture cmp_bhv;
```

7.6 MUX

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity MUX is
8  port(
9          hr0_in: in std_logic_vector (3 downto 0);
10         hr1_in: in std_logic_vector (1 downto 0);
11         min0_in: in std_logic_vector (3 downto 0);
12         min1_in: in std_logic_vector (2 downto 0);
13         s0_in: in std_LOGIC_vector(3 downto 0);
14         s1_in: in std_LOGIC_vector(2 downto 0);
15
16         hr0_in_alarm: in std_logic_vector (3 downto 0);
17         hr1_in_alarm: in std_logic_vector (1 downto 0);
18         min0_in_alarm: in std_logic_vector (3 downto 0);
19         min1_in_alarm: in std_logic_vector (2 downto 0);
20
21         seg1_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
22         seg2_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
23         seg3_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
24         seg4_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
25         seg5_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
26         seg6_alarm : OUT STD_LOGIC_VECTOR(6 downto 0);
27
28         mode_in_mux: in std_logic
29
30         );
31 end entity MUX;
32
33 architecture bhv_mux of MUX is
34     signal sls1 :   STD_LOGIC_VECTOR(3 DOWNTO 0);
35     signal sms1 :   STD_LOGIC_VECTOR(2 DOWNTO 0);
36     signal mls1 :   STD_LOGIC_VECTOR(3 DOWNTO 0);
37     signal mms1 :   STD_LOGIC_VECTOR(2 DOWNTO 0);
38     signal hls1 :   STD_LOGIC_VECTOR(3 DOWNTO 0);
39     signal hms1 :   STD_LOGIC_VECTOR(1 DOWNTO 0);
40
41
42     begin
43
44 process(hr0_in, hr1_in, min0_in, min1_in, s0_in, s1_in, min1_in_alarm,
       min0_in_alarm, hr1_in_alarm, hr0_in_alarm, mode_in_mux) begin
45
46     if (mode_in_mux='0') then --"mode" for clock
47
48         hms1<=hr1_in;
49         hls1<=hr0_in;
50         mms1<=min1_in;
51         mls1<=min0_in;
52         sms1<=s1_in;
53         sls1<=s0_in;
54
55     else                --"mode" for alarm
56
57         hms1<=hr1_in_alarm;
58         hls1<=hr0_in_alarm;
59         mms1<=min1_in_alarm;
60         mls1<=min0_in_alarm;
61
62     end if;
63
64 end process;
65
66 PROCESS (sms1)
67         BEGIN
68             CASE sms1 IS
69                 WHEN "000" => seg2_alarm <= "0000001";
70                 WHEN "001" => seg2_alarm <= "1001111";
```

```
71                  WHEN "010" => seg2_alarm <= "0010010";
72                  WHEN "011" => seg2_alarm <= "0000110";
73                  WHEN "100" => seg2_alarm <= "1001100";
74                  WHEN "101" => seg2_alarm <= "0100100";
75                  WHEN OTHERS => seg2_alarm <= "0000001";
76              END CASE;
77      END PROCESS;
78      -- For the LS of the second
79      PROCESS (sls1)
80          BEGIN
81              CASE sls1 IS
82                   WHEN "0000" => seg1_alarm <= "0000001";
83                   WHEN "0001" => seg1_alarm <= "1001111";
84                   WHEN "0010" => seg1_alarm <= "0010010";
85                   WHEN "0011" => seg1_alarm <= "0000110";
86                   WHEN "0100" => seg1_alarm <= "1001100";
87                   WHEN "0101" => seg1_alarm <= "0100100";
88                   WHEN "0110" => seg1_alarm <= "0100000";
89                   WHEN "0111" => seg1_alarm <= "0001111";
90                   WHEN "1000" => seg1_alarm <= "0000000";
91                   WHEN "1001" => seg1_alarm <= "0000100";
92                   WHEN OTHERS => seg1_alarm <= "0000001";
93              END CASE;
94      END PROCESS;
95
96      --- For the MS of the Second
97      PROCESS (mms1)
98          BEGIN
99              CASE mms1 IS
100                 WHEN "000" => seg4_alarm <= "0000001";
101                 WHEN "001" => seg4_alarm <= "1001111";
102                 WHEN "010" => seg4_alarm <= "0010010";
103                 WHEN "011" => seg4_alarm <= "0000110";
104                 WHEN "100" => seg4_alarm <= "1001100";
105                 WHEN "101" => seg4_alarm <= "0100100";
106                 WHEN OTHERS => seg4_alarm <= "0000001";
107             END CASE;
108     END PROCESS;
109
110     -- For the LS of the second
111     PROCESS (mls1)
112         BEGIN
113         CASE mls1 IS
114              WHEN "0000" => seg3_alarm <= "0000001";
115              WHEN "0001" => seg3_alarm <= "1001111";
116              WHEN "0010" => seg3_alarm <= "0010010";
117              WHEN "0011" => seg3_alarm <= "0000110";
118              WHEN "0100" => seg3_alarm <= "1001100";
119              WHEN "0101" => seg3_alarm <= "0100100";
120              WHEN "0110" => seg3_alarm <= "0100000";
121              WHEN "0111" => seg3_alarm <= "0001111";
122              WHEN "1000" => seg3_alarm <= "0000000";
123              WHEN "1001" => seg3_alarm <= "0000100";
124              WHEN OTHERS => seg3_alarm <= "0000001";
125          END CASE;
126     END PROCESS;
127
128     --For the MS of the hour hand
129     PROCESS(hms1)
130         BEGIN
131             CASE hms1 IS
132                 WHEN "00" => seg6_alarm <= "0000001";
133                 WHEN "01" => seg6_alarm <= "1001111";
134                 WHEN "10" => seg6_alarm <= "0010010";
135                 WHEN OTHERS => seg6_alarm <= "0000001";
136             END CASE;
137     END PROCESS;
138
139     -- For the LS of the hour hand
140     PROCESS(hls1)
141         BEGIN
142             CASE hls1 IS
143              WHEN "0000" => seg5_alarm <= "0000001";
144              WHEN "0001" => seg5_alarm <= "1001111";
```

```
145             WHEN "0010" => seg5_alarm <= "0010010";
146             WHEN "0011" => seg5_alarm <= "0000110";
147             WHEN "0100" => seg5_alarm <= "1001100";
148             WHEN "0101" => seg5_alarm <= "0100100";
149             WHEN "0110" => seg5_alarm <= "0100000";
150             WHEN "0111" => seg5_alarm <= "0001111";
151             WHEN "1000" => seg5_alarm <= "0000000";
152             WHEN "1001" => seg5_alarm <= "0000100";
153             WHEN OTHERS => seg5_alarm <= "0000001";
154           END CASE;
155     END PROCESS;
156
157 end architecture bhv_mux;
```

8 References

References

Clock Devider
https://www.codeproject.com/Tips/444385/Frequency-Divider-with-VHDL-2

Alarm Counter
https://vhdlguru.blogspot.com/2010/03/digital-clock-in-vhdl.html

State Machines
https://www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine

Various findings in Stackoverflow
https://stackoverflow.com/

VHDL for LCD
https://www.digikey.com/eewiki/pages/viewpage.action?pageId=4096079

UPPSALA
UNIVERSITET