# EEL 3701 – Digital Logic and Computer Systems Lab 4

# Greg Bolling

# UF ID 64911960

## Problem Statement

The goal of the lab is to design, develop, and implement using VHDL/Quartus, a controller for an alarm clock on the FPGA and extension board. The extension board has four seven segment displays for displaying the time in 24-hour mode. The circuit also implements an alarm feature that will set off a light and buzzer to alert the owner that the time has been reached. The alarm and clock are both able to be set using switches to increment the hours / minutes when in the time set or alarm set modes.

## Design

The alarm and clock are driven by a state machine that keeps track of four possible states:

State: Time Meaning: Running normally, waiting to see when/if an alarm condition is met.

State: Alarm Meaning: An alarm has been reached and is going off.

State: Set Time Meaning: Time is being set by the user, advancing minutes or hours

State: Set\_Alarm Meaning: Alarm is being set by the user, advancing minutes or hours

There is a counter that handles the counting from 0..23 and back to 0 in the hours and a counter that handles the 0..59 and back to 0 for the minutes. This counter has special conditions when it is running from 0.1.2 in the hours and back to 0 as opposed to the minutes that count 0,1,2..5 and back to 0. It also tests the seconds to know when it should change the hours and minutes.

The state machine uses a set of blocks in the design:

TwoDigitDisplay This module takes in two digits, in BCD format and displays them on two seven segment displays.

ClockCounter This module is responsible for generating the right next count for either the alarm or the time. If it’s the time, this has an enable signal to allow the time to increment. If it’s the alarm, it only increments using the fast increment inputs. The time has the same fast increment inputs to allow it to be set. This is used twice in the design. Once to create Alarm and once to create Time.

FSM This is the finite state machine that drives the entire clock. It has four states as described above and has a transition diagram as shown below.

SecondMux This is a mux function that takes the four seven segment inputs (that are either the time or the alarm) and converts them to digit displays that are sequenced one after another to the display using the segment selects and the seven segment values (A,B,C,D,E,F,G). It also takes in the control signals from the FSM and outputs the alarm / lights. The output is borrowed from a previous lab.

Slow\_Clock This is a process that creates a 1 Hz clock.

## Implementation

All of the designs were implemented using VHDL and similar to previous labs. The slow clock was used most of the time (labeled Clk1Hz.) This was because its different than the other clock called clock and runs at 4 MHz.

The state machine has the following inputs:

clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

Clk1Hz : in std\_logic;

Switch1 : in std\_logic; -- "Set Time" mode

Switch2 : in std\_logic; -- "Set Alarm" mode

Switch3 : in std\_logic; -- Minute Increment

Switch4 : in std\_logic; -- Hours Increment

Switch5 : in std\_logic; -- Activates Alarm / Enable

Switch7 : in std\_logic; -- Mutes the Alarm, it never rings

PushButton1 : in std\_logic;

AlarmEqualsTimeFSM : in std\_logic;

The Alarm Equals Time FSM is true (high) when the alarm and time have the same BCD values.

The state machine has the following outputs:

FastMinTime : out std\_logic;

FastMinAlarm : out std\_logic;

FastHourTime : out std\_logic;

FastHourAlarm : out std\_logic;

CntEn : out std\_logic;

SelectTime\_ALn : out std\_logic;

SoundAlarm : out std\_logic;

SetLED0Off : out std\_logic;

SetLED0On : out std\_logic;

SetLed04Sec75Duty : out std\_logic;

SsetLED2Sec50Duty : out std\_logic;

Led1Out : out std\_logic

FastMinTime and FasMinAlarm set the time or alarm minutes at once per second.

FastHourTime and FasHourAlarm set the time or alarm hours at once per second.

CntEn is if the clock counter should count or not. It won’t count when it’s stopped.

Select Time ALn is high when the time is to be displayed, and low when the alarm is to be displayed

Sound Alarm sets off the alarm

SetLED0Off turns LED0 off.

SetLED0On turns LED0 on.

SetLED04Sec75Duty sets a 4 second, 75% duty cycle to LED 0

SetLED2Sec450Duty sets a 2 second, 50% duty cycle to LED 0

Led1Out is high when on, low when off.

SecondMux handles the seven segment display rotation and it also takes in all the inputs for LED0 and LED1 and turns them into the right lights for the LED. It also has the ability to drive the Alarm in a cycle as opposed to always on. It also has two counters inside it that create the 50% duty cycle and the 75% duty cycle for the outputs.

I created a top level VHDL file called Lab04 that puts the blocks together.

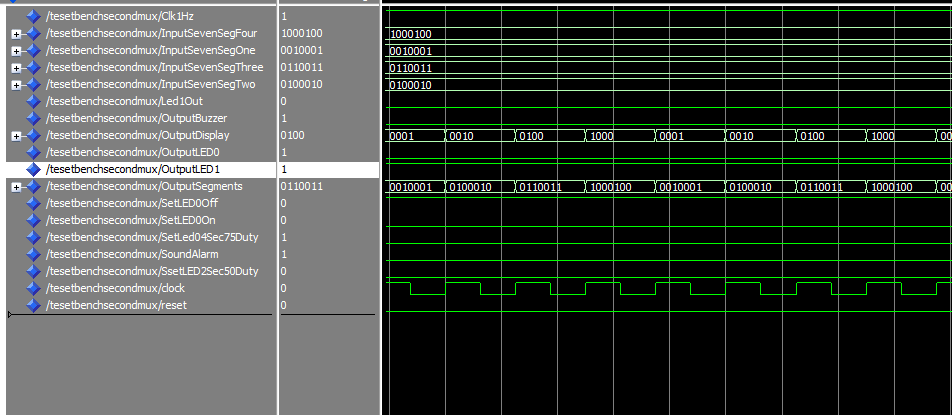
## Conclusions

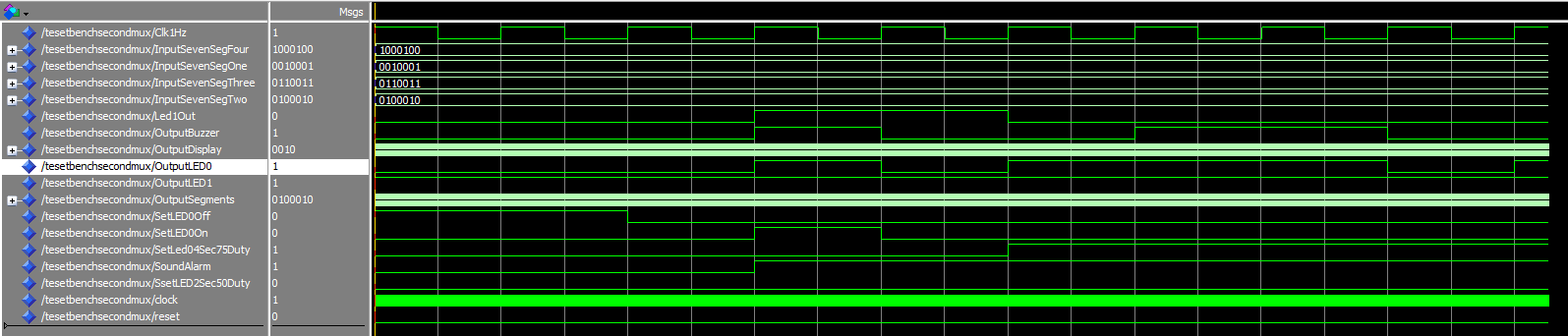
VHDL’s made creating this a lot easier than using truth tables and equations. I think that it will make it easier to change and reminds me of the code for programming 1 and 2. There is a difference and it seems like the testbench concept was a bit difficult. I had to use a lot of resources to find out what a testbench was and how it worked. The clock was the hardest because it doesn’t seem like process can do that but it also sounds like that’s how VHDL works. I also had to figure out how to create signal inputs for reset and switches and the BCD values that changed in time.

Appendix

Picture(s)

SecondMux Simulation





Two Digit Display Simulation



VHDL Files

Two Digit Display Module

---------------------------------------

-- Author: Greg Bolling

-- Date: February 3, 2020

-- Class; EEL 3701

-- Assignment: Lab 02 svnseg

-- Class: 12368

-- Section 7441

-- PI Name: Savvas Ferekides

---------------------------------------

library IEEE;

use IEEE.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity TwoDigitDisplay is

port

(

binary1 : in std\_logic\_vector(3 downto 0);

binary2 : in std\_logic\_vector(3 downto 0);

a\_to\_g\_1 : out std\_logic\_vector(6 downto 0);

a\_to\_g\_2 : out std\_logic\_vector(6 downto 0)

);

end TwoDigitDisplay;

architecture arch\_TwoDigitDisplay of TwoDigitDisplay is

begin

P1: process(binary1)

begin

case (binary1) is

when "0000" => a\_to\_g\_1 <= "1000000"; -- display 0 => '1' is off and a '0' is on

when "0001" => a\_to\_g\_1 <= "1111001"; -- display 1 => '1' is off and a '0' is on

when "0010" => a\_to\_g\_1 <= "0100100"; -- display 2 => '1' is off and a '0' is on

when "0011" => a\_to\_g\_1 <= "0110000"; -- display 3 => '1' is off and a '0' is on

when "0100" => a\_to\_g\_1 <= "0011001"; -- display 4 => '1' is off and a '0' is on

when "0101" => a\_to\_g\_1 <= "0010010"; -- display 5 => '1' is off and a '0' is on

when "0110" => a\_to\_g\_1 <= "0000010"; -- display 6 => '1' is off and a '0' is on

when "0111" => a\_to\_g\_1 <= "1111000"; -- display 7 => '1' is off and a '0' is on

when "1000" => a\_to\_g\_1 <= "0000000"; -- display 8 => '1' is off and a '0' is on

when "1001" => a\_to\_g\_1 <= "0011000"; -- display 9 => '1' is off and a '0' is on

when others => a\_to\_g\_1 <= "1111111"; -- display nothing

end case;

end process;

P2: process(binary2)

begin

case (binary2) is

when "0000" => a\_to\_g\_2 <= "1000000"; -- display 0 => '1' is off and a '0' is on

when "0001" => a\_to\_g\_2 <= "1111001"; -- display 1 => '1' is off and a '0' is on

when "0010" => a\_to\_g\_2 <= "0100100"; -- display 2 => '1' is off and a '0' is on

when "0011" => a\_to\_g\_2 <= "0110000"; -- display 3 => '1' is off and a '0' is on

when "0100" => a\_to\_g\_2 <= "0011001"; -- display 4 => '1' is off and a '0' is on

when "0101" => a\_to\_g\_2 <= "0010010"; -- display 5 => '1' is off and a '0' is on

when "0110" => a\_to\_g\_2 <= "0000010"; -- display 6 => '1' is off and a '0' is on

when "0111" => a\_to\_g\_2 <= "1111000"; -- display 7 => '1' is off and a '0' is on

when "1000" => a\_to\_g\_2 <= "0000000"; -- display 8 => '1' is off and a '0' is on

when "1001" => a\_to\_g\_2 <= "0011000"; -- display 9 => '1' is off and a '0' is on

when others => a\_to\_g\_2 <= "1111111"; -- display nothing

end case;

end process;

end arch\_TwoDigitDisplay;

SecondMux (this is the MUX and Alarm Controller VHDL)

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

--------------fastclock for sevensegment display-------------

entity SecondMux is

port(

clock : in std\_logic;

Clk1Hz : in std\_logic;

reset : in std\_logic;

SoundAlarm : in std\_logic;

SetLED0Off : in std\_logic;

SetLED0On : in std\_logic;

SetLed04Sec75Duty : in std\_logic;

SsetLED2Sec50Duty : in std\_logic;

Led1Out : in std\_logic;

InputSevenSegOne : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegTwo : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegThree : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegFour : in std\_logic\_vector(6 DOWNTO 0);

OutputLED0 : out std\_logic;

OutputLED1 : out std\_logic;

OutputBuzzer : out std\_logic;

OutputDisplay : out std\_logic\_vector(3 downto 0);

OutputSegments : out std\_logic\_vector(6 DOWNTO 0)

);

end SecondMux;

architecture Behavioral of SecondMux is

type my\_fms\_display\_states is (S0, S1, S2, S3);

signal present\_state, next\_state : my\_fms\_display\_states;

-- below 2s and 4s counters are copied from template provided in canvas

signal clock\_2seconds: std\_logic:='0'; -- clock thick after every 2 seconds

signal clock\_4seconds: std\_logic:='0'; -- clock thick after every 4 seconds

signal counter2s : std\_logic\_vector(1 DOWNTO 0); --counter for 2s clock period with 50% duty cycle

signal counter4s : std\_logic\_vector(1 DOWNTO 0); --counter for 4s clock period with 75% duty cycle

begin

OutputLED0 <= clock\_4seconds when (SetLed04Sec75Duty = '1') else

clock\_2seconds when (SsetLED2Sec50Duty = '1') else

'1' when (SetLED0On = '1')

else '0';

OutputLED1 <= not reset;

OutputBuzzer <= clock\_2seconds when ((SoundAlarm = '1') and (reset = '0')) else '0';

SevenSegRegister : process(clock, reset)

begin

if(reset = '1') then

present\_state <= S0;

elsif (clock='1' and clock'event) then

present\_state <= next\_state;

end if;

end process;

SevenSegCount : process(present\_state, InputSevenSegOne, InputSevenSegTwo, InputSevenSegThree, InputSevenSegFour)

begin

case present\_state is

when S0 =>

next\_state <= S1;

OutputSegments<= InputSevenSegOne;

OutputDisplay <= "000" & not reset;

when S1 =>

next\_state <= S2;

OutputSegments<= InputSevenSegTwo;

OutputDisplay <= "0010";

when S2 =>

next\_state <= S3;

OutputSegments<= InputSevenSegThree;

OutputDisplay <= "0100";

when S3 =>

next\_state <= S0;

OutputSegments<= InputSevenSegFour;

OutputDisplay <= "1000";

end case;

end process;

-- below 2s and 4s counters are copied from template provided in canvas

TwoSecProc : process (Clk1Hz, reset) --clock of 2s period 50% duty cycle

begin

if(reset= '1') then

counter2s <= (others=>'0');

clock\_2seconds <= '0';

elsif (Clk1Hz='1' and Clk1Hz'event) then

if(counter2s >= "01") then

counter2s <= (others=>'0');

clock\_2seconds <= not clock\_2seconds;

else

counter2s <= counter2s + 1;

end if;

end if;

end process;

-- below 2s and 4s counters are copied from template provided in canvas

FourSecProc : process (Clk1Hz, reset) --clock of 4s period; 75% duty cycle

begin

if(reset= '1') then

counter4s <= (others=>'0');

clock\_4seconds <= '0';

elsif (Clk1Hz='1' and Clk1Hz'event) then

if(counter4s >= "11") then

counter4s <= (others=>'0');

else

counter4s <= counter4s + 1;

end if;

if(counter4s = "00" or counter4s = "01" or counter4s = "10") then

clock\_4seconds <= '1';

else

clock\_4seconds <= '0';

end if;

end if;

end process;

end Behavioral;

Testbench Signal Generation Idea

I am crediting the source for generating a clock and signals inside a test bench.

To create a clock, I had to look this up as I didn’t know how to do it. I found two good sources at

<https://www.eng.auburn.edu/~nelsovp/courses/elec4200/Slides/VHDL%206%20Testbench.pdf>

for showing how to create a clock process. Slide 7 showed wait for period T1 and T2 which were the pulse time periods of the clock. I also found this same example at

<https://electronics.stackexchange.com/questions/148320/proper-clock-generation-for-vhdl-testbenches>

so it looked like it would work and it did. I then used the reference to the after command to change signals after a time period to create the rest of the inputs that were not clocks like reset, pushbutton and switch inputs.

Testbench for Second Mux

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

--------------fastclock for sevensegment display-------------

entity TesetbenchSecondMux is

end TesetbenchSecondMux;

architecture Behavioral of TesetbenchSecondMux is

signal clock : std\_logic;

signal Clk1Hz : std\_logic;

signal reset : std\_logic;

signal SoundAlarm : std\_logic;

signal SetLED0Off : std\_logic;

signal SetLED0On : std\_logic;

signal SetLed04Sec75Duty : std\_logic;

signal SsetLED2Sec50Duty : std\_logic;

signal Led1Out : std\_logic;

signal InputSevenSegOne : std\_logic\_vector(6 DOWNTO 0);

signal InputSevenSegTwo : std\_logic\_vector(6 DOWNTO 0);

signal InputSevenSegThree : std\_logic\_vector(6 DOWNTO 0);

signal InputSevenSegFour : std\_logic\_vector(6 DOWNTO 0);

signal OutputLED0 : std\_logic;

signal OutputLED1 : std\_logic;

signal OutputBuzzer : std\_logic;

signal OutputDisplay : std\_logic\_vector(3 downto 0);

signal OutputSegments : std\_logic\_vector(6 DOWNTO 0);

Component SecondMux

port(

clock : in std\_logic;

Clk1Hz : in std\_logic;

reset : in std\_logic;

SoundAlarm : in std\_logic;

SetLED0Off : in std\_logic;

SetLED0On : in std\_logic;

SetLed04Sec75Duty : in std\_logic;

SsetLED2Sec50Duty : in std\_logic;

Led1Out : in std\_logic;

InputSevenSegOne : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegTwo : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegThree : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegFour : in std\_logic\_vector(6 DOWNTO 0);

OutputLED0 : out std\_logic;

OutputLED1 : out std\_logic;

OutputBuzzer : out std\_logic;

OutputDisplay : out std\_logic\_vector(3 downto 0);

OutputSegments : out std\_logic\_vector(6 DOWNTO 0)

);

end component;

begin

Part1 : SecondMux

port map (

clock => clock , -- : in std\_logic;

Clk1Hz => Clk1Hz , -- : in std\_logic;

reset => reset , -- : in std\_logic;

SoundAlarm => SoundAlarm , -- : in std\_logic;

SetLED0Off => SetLED0Off , -- : in std\_logic;

SetLED0On => SetLED0On , -- : in std\_logic;

SetLed04Sec75Duty => SetLed04Sec75Duty , -- : in std\_logic;

SsetLED2Sec50Duty => SsetLED2Sec50Duty , -- : in std\_logic;

Led1Out => Led1Out , -- : in std\_logic;

InputSevenSegOne => InputSevenSegOne , -- : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegTwo => InputSevenSegTwo , -- : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegThree => InputSevenSegThree, -- : in std\_logic\_vector(6 DOWNTO 0);

InputSevenSegFour => InputSevenSegFour , -- : in std\_logic\_vector(6 DOWNTO 0);

OutputLED0 => OutputLED0 , -- : out std\_logic;

OutputLED1 => OutputLED1 , -- : out std\_logic;

OutputBuzzer => OutputBuzzer , -- : out std\_logic;

OutputDisplay => OutputDisplay , -- : out std\_logic\_vector(3 downto 0);

OutputSegments => OutputSegments -- : out std\_logic\_vector(6 DOWNTO 0)

);

P0: process

begin

clock <= '1';

wait for 125 ns;

clock <= '0';

wait for 125 ns;

end process;

P1: process

begin

Clk1Hz <= '1';

wait for 500 ms;

Clk1Hz <= '0';

wait for 500 ms;

end process;

reset <= '1', '0' after 2000 ns;

SoundAlarm <= '0', '1' after 3000 ms;

SetLED0Off <= '1', '0' after 2000 ms;

SetLED0On <= '0', '1' after 3000 ms, '0' after 4000 ms;

SetLed04Sec75Duty <= '0', '1' after 5000 ms, '0' after 15000 ms;

SsetLED2Sec50Duty <= '0', '1' after 16000 ms, '0' after 20000 ms;

Led1Out <= '0', '1' after 3000 ms, '0' after 5000 ms;

InputSevenSegOne <= "0010001";

InputSevenSegTwo <= "0100010";

InputSevenSegThree <= "0110011";

InputSevenSegFour <= "1000100";

end Behavioral;

Testbench for Two Digit Seven Segment

---------------------------------------

-- Author: Greg Bolling

-- Date: February 3, 2020

-- Class; EEL 3701

-- Assignment: Lab 02 svnseg

-- Class: 12368

-- Section 7441

-- PI Name: Savvas Ferekides

---------------------------------------

library IEEE;

use IEEE.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use IEEE.std\_logic\_unsigned.all;

entity TestbenchTwoDigitDisplay is

end TestbenchTwoDigitDisplay;

architecture arch\_TwoDigitDisplay of TestbenchTwoDigitDisplay is

Component TwoDigitDisplay

port

(

binary1 : in std\_logic\_vector(3 downto 0);

binary2 : in std\_logic\_vector(3 downto 0);

a\_to\_g\_1 : out std\_logic\_vector(6 downto 0);

a\_to\_g\_2 : out std\_logic\_vector(6 downto 0)

);

end component;

signal binary1 : std\_logic\_vector(3 downto 0);

signal binary2 : std\_logic\_vector(3 downto 0);

signal a\_to\_g\_1 : std\_logic\_vector(6 downto 0);

signal a\_to\_g\_2 : std\_logic\_vector(6 downto 0);

begin

TwoDigit: TwoDigitDisplay

port map

(

binary1 => binary1 , -- : in std\_logic\_vector(3 downto 0);

binary2 => binary2 , -- : in std\_logic\_vector(3 downto 0);

a\_to\_g\_1 => a\_to\_g\_1, -- : out std\_logic\_vector(6 downto 0);

a\_to\_g\_2 => a\_to\_g\_2 -- : out std\_logic\_vector(6 downto 0)

);

binary1 <= "0000", "0001" after 1000 ns,

"0010" after 2000 ns,

"0011" after 3000 ns,

"0100" after 4000 ns,

"0101" after 5000 ns,

"0110" after 6000 ns,

"0111" after 7000 ns,

"1000" after 8000 ns,

"1001" after 9000 ns,

"1010" after 10000 ns,

"1011" after 11000 ns,

"1100" after 12000 ns,

"1101" after 13000 ns,

"1110" after 14000 ns,

"1111" after 15000 ns;

binary2 <= "0000", "0001" after 1000 ns,

"0010" after 2000 ns,

"0011" after 3000 ns,

"0100" after 4000 ns,

"0101" after 5000 ns,

"0110" after 6000 ns,

"0111" after 7000 ns,

"1000" after 8000 ns,

"1001" after 9000 ns,

"1010" after 10000 ns,

"1011" after 11000 ns,

"1100" after 12000 ns,

"1101" after 13000 ns,

"1110" after 14000 ns,

"1111" after 15000 ns;

end arch\_TwoDigitDisplay;

