# EEL 3701 – Digital Logic and Computer Systems Lab 3

# Greg Bolling

# UF ID 64911960

## Problem Statement

The goal of the lab is to build sequential circuits from memory elements. In the first problem, an alarm system is synthesized from an SR-Latch. The design of the alarm system is simple to have a set and reset function.

## Design

Problem 1 Part 1 starts with creating an alarm system with a panel button that resets the alarm (off state) and a door switch that can turn the alarm on (on-state.) The SR Latch was chosen because it has Set and Reset inputs that align to the function of turning an alarm on and turning an alarm off. The latch was chosen over combinatorial logic because a latch holds the memory of the last event.

SR-Latch

Set is connected to the door switch. When that goes high, the alarm sounds. Connect the switch to a dual position switch (alternatively could use a pulldown resistor and a pushbutton switch to Vcc.)

Reset is connected at the panel to turn the alarm off. It is just connected to Vcc when the button is pushed.

The S-R Latch is created in VHDL using two key combinatorial logic gates:

Q <= R nor notQ;

notQ <= S nor Q;

I will use these pins:

S Pin J7

R Pin F4

Q Pin N10 (Alarm / LED)

The design could have used a JK Latch to handle the case when the door is opened and the alarm is being reset when they both close but this is a simple design so maybe it is not necessary. The design had warnings but no errors in Quartus.

Problem 2

I implemented the code provided in the pre-lab. This is a shift register that clocks all 1’s (LED on) into the register and shifts in a 0 over and over until the entire display is all 0 (LED’s off) and then starts over by setting it to all 1’s (LED on).

I will use these pins:

clock Pin H4

LED[7] Pin N10

LED[6] Pin N12

LED[5] Pin M13

LED[4] Pin L13

LED[3] Pin J13

LED[2] Pin H13

LED[1] Pin G13

LED[0] Pin K7

Reset Pin G4

I reversed this shift direction by inserting a ‘0’ at bit 0 and then shifting everything to bits 7 down to 1. The code for this was modified as :

shift\_reg(7) <= '0';

shift\_reg(6 downto 0) <= shift\_reg(7 downto 1);

I then modified the design to do a left shift followed by a right shift indefinitely. To do this, I set a register to 0 to show that I haven’t started to right shift. I then set the register to a 1 and that tells me to do right shifts as opposed to left shifts. The system works the same otherwise. I used the same pin assignments from the shift registers above.

Problem 3

I modified the provided VHDL code to add the state machine transitions in VHDL as follows:

P1: process(present\_state, a\_g)

begin

case present\_state is

when S0 =>

if(a\_g = "0000110") then

next\_state <= S1;

else

next\_state <= S0;

end if;

a\_g<= "0000110";

display <= "0001";

when S1 =>

if(a\_g = "0100100") then

next\_state <= S2;

else

next\_state <= S1;

end if;

a\_g<= "0100100";

display <= "0010";

when S2 =>

if(a\_g = "0000000") then

next\_state <= S2;

else

next\_state <= S1;

end if;

a\_g<= "0000000";

display <= "0100";

when S3 =>

if(a\_g = "0001100") then

next\_state <= S2;

else

next\_state <= S1;

end if;

a\_g<= "0001100";

display <= "1000";

end case;

end process;

Pin Assignments for the seven segment display (I renamed LED to SevenSeg because I used LED above and it became a problem keeping this aligned.)

clock Pin H4

reset Pin G4

SevenSeg[6] Pin D11

SevenSeg[5] Pin M12

SevenSeg[4] Pin M11

SevenSeg[3] Pin M10

SevenSeg[2] Pin M9

SevenSeg[1] Pin K12

SevenSeg[0] Pin K8

Display[3] Pin H3

Display[2] Pin H2

Display[1] Pin K5

Display[0] Pin J2

Problem 4

The code for counting 0,1,3,5,7,9,0 is below in this file. It will use the same pin assignments as in Problem 3 last part above.

clock Pin H4

reset Pin G4

SevenSeg[6] Pin D11

SevenSeg[5] Pin M12

SevenSeg[4] Pin M11

SevenSeg[3] Pin M10

SevenSeg[2] Pin M9

SevenSeg[1] Pin K12

SevenSeg[0] Pin K8

Display[3] Pin H3

Display[2] Pin H2

Display[1] Pin K5

Display[0] Pin J2

## Implementation

## Conclusions

Appendix

Picture(s)

VHDL Files

Problem 4

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

use ieee.std\_logic\_unsigned.all;

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

entity lab3part4 is

Port (clock, reset : in STD\_LOGIC;

display: out std\_logic\_vector(3 DOWNTO 0);

SevenSeg : out std\_logic\_vector(6 DOWNTO 0));

end lab3part4;

architecture Behavioral of lab3part4 is

signal a\_g : std\_logic\_vector(6 downto 0); -- declare the (state-machine) enumerated type

signal count : std\_logic\_vector(3 downto 0); -- counter

signal slow\_clk : std\_logic;

component slow\_clock is

port(

clock, reset : in STD\_LOGIC;

slow\_clk : out STD\_LOGIC); end component;

begin

U0: slow\_clock

port map(

slow\_clk=>slow\_clk,

reset=>reset,

clock=>clock);

P0: process(slow\_clk, reset)

begin

if(reset = '1') then

count <= "0000";

elsif (slow\_clk ='1' and slow\_clk'event) then

case (count) is

when "0000" => count <= "0011";

when "0011" => count <= "0101";

when "0101" => count <= "0111";

when "0111" => count <= "1001";

when "1001" => count <= "0000";

when others => count <= "0000";

end case;

end if;

end process;

SevenSeg <= a\_g;

end Behavioral;