Due on 12/8/2018

**EEL4740**

**Embedded Systems**

Project Title: Vending Machine

Author:

Name: Italo Peirano PID: 5660995



Florida International University

**Project Objectives:**

Develop a Vending machine using the board Elbert V2 – Spartan 3A, and its internal components. The project uses a single display that shows the price of the 4 different beverages that are on sale. 4 led positions that show where each different types of beverage are located. 3 switches that represent the different coins inserted into the vending machine. A reset button to go back to initial conditions at any moment, in this case is switch 6. A start button (switch 4) that allows a person to select the different drinks available. A back button that allows the user to make another selection after a drink has already being selected. 3 single pole switches that are used for the selection of 4 beverages.

**Description**:

The vending machine consist of 4 different types of drinks ( Pepsi, 7Up, Fanta, Coke) the drinks are located horizontally in 4 columns. When coke is dispensed then led1 lights up, when fanta is dispensed then led2 lights up, when 7up is dispensed then led3 lights up, and when Pepsi is dispensed led4 lights up. To select the different drinks with the single pole switches from 000 selects coke, 001 selects fanta, 010 selects 7Up, 011 selects Pepsi.

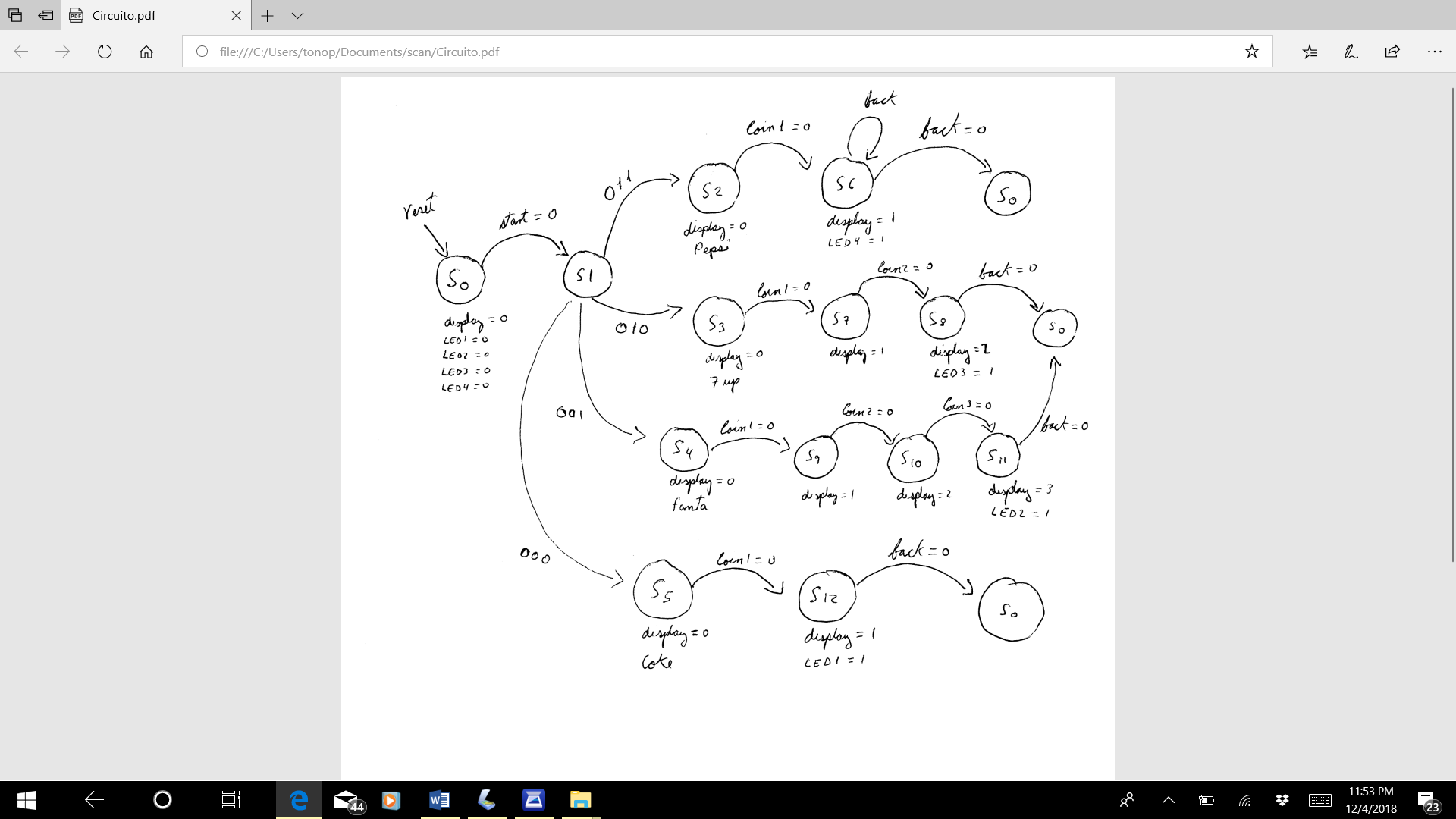
All drinks have different prices, and when you insert a coin the display shows the amount inserted so far.

The prices are as follows: coke is $1, Fanta is $3, 7Up is $2, Pepsi is $1.

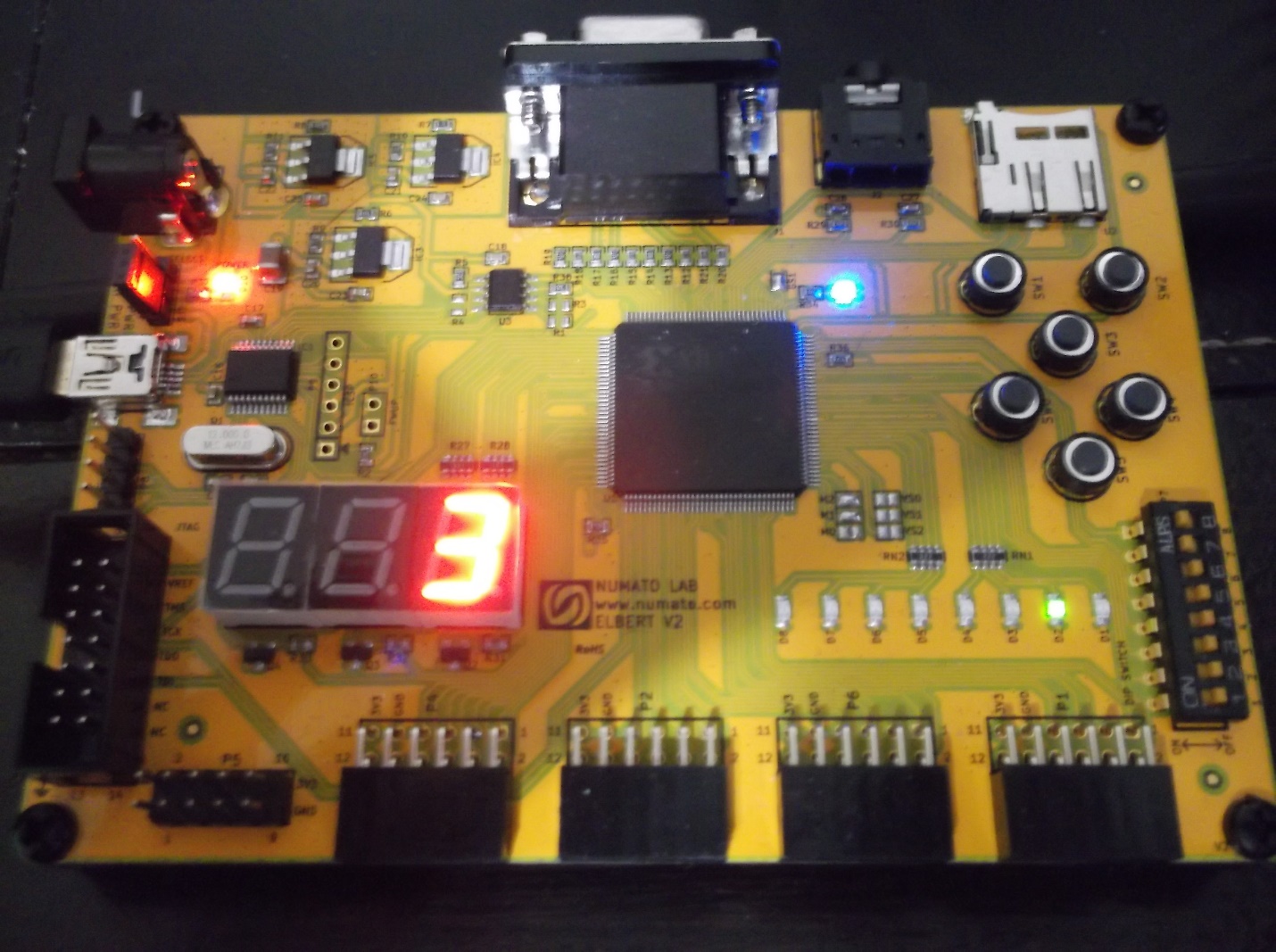
**Components Used:**

FPGA board ( one bcd display, 4 leds, 6 push buttons, 3 single pole switches)

**Moore schematic:**



**Picture of Vending machine :**

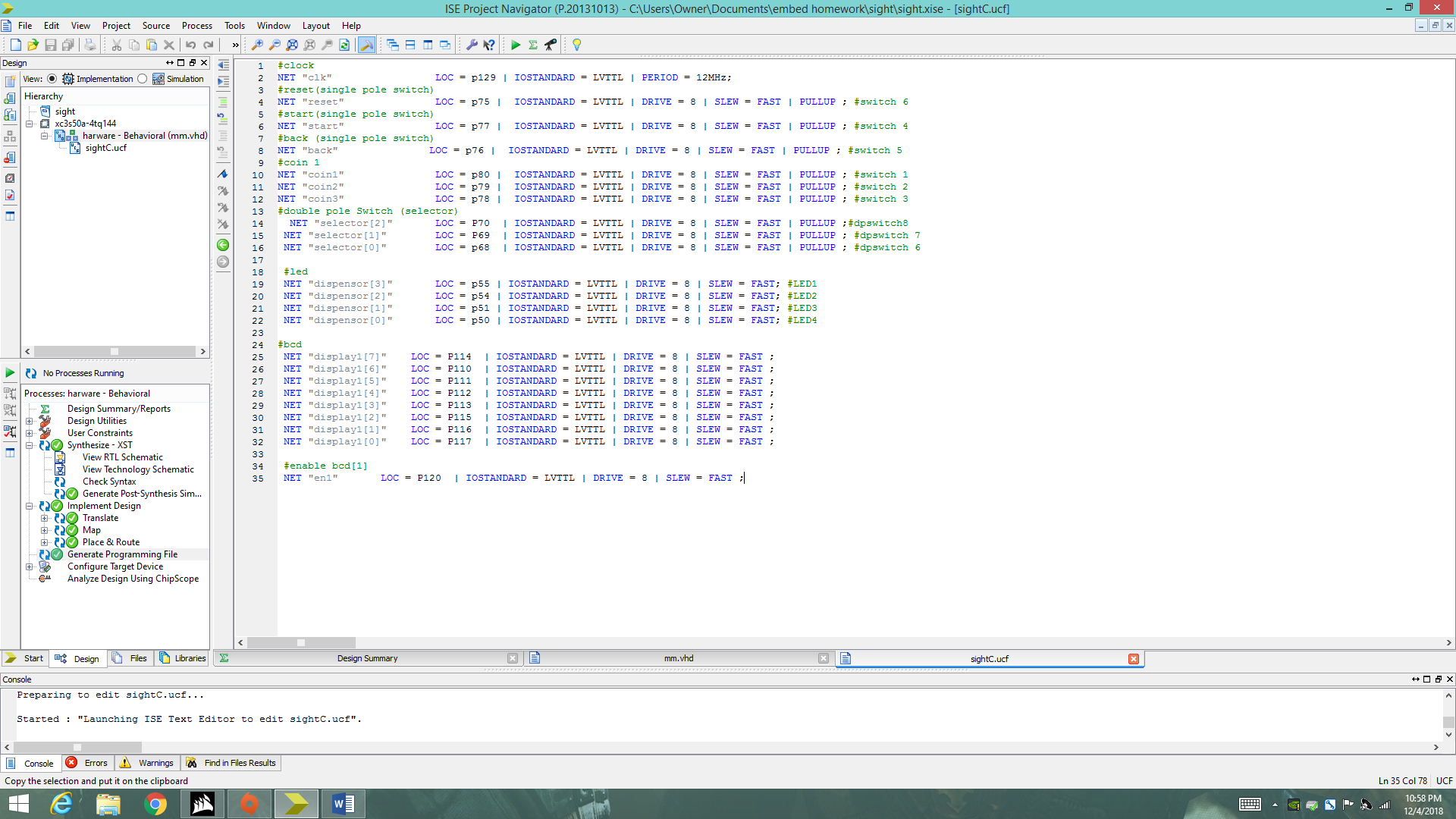


In this picture we can see a price of 3 dollars and Fanta drink being dispensed because led2 is on.

**Project Video Link:**

[**https://drive.google.com/file/d/1kD2gu-GVRsPXiSmNDcMWNQ1x6yFwY0D0/view?usp=sharing**](https://drive.google.com/file/d/1kD2gu-GVRsPXiSmNDcMWNQ1x6yFwY0D0/view?usp=sharing)

**Constraint Code:**



**VHDL Code:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity harware is

port( clk : in STD\_LOGIC;

start: in STD\_LOGIC;

coin1, coin2, coin3 : in STD\_LOGIC;

reset : in STD\_LOGIC;

back : in STD\_LOGIC;

selector : in STD\_LOGIC\_VECTOR (2 downto 0);

dispensor: out STD\_LOGIC\_vector (3 downto 0);

en1: out STD\_LOGIC;

display1 : out STD\_LOGIC\_VECTOR (7 downto 0)

) ;

end harware ;

architecture Behavioral of harware is

type statetype is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12);

signal pstate,nstate : statetype;

begin

en1 <= '0';

synchronous\_process: process(clk)

begin

if rising\_edge (clk) then

if(reset = '0' ) then

pstate <= s0;

else

pstate <= nstate;

end if;

end if;

end process;

Output\_decoder: process (pstate, start, coin1, coin2, coin3, back, selector)

begin

nstate <= s0;

case(pstate) is

when s0 =>

if(start = '0') then nstate <= s1;

elsif(start = '1') then nstate <= s0;

end if;

when s1 =>

if(selector = "011") then nstate <= s2;

elsif(selector = "010") then nstate <= s3;

elsif(selector = "001") then nstate <= s4;

elsif(selector = "000") then nstate <= s5;

end if;

when s2 =>

if(coin1 = '0') then nstate <= s6;

else

nstate <= s2;

end if;

when S6 =>

if(back = '0') then nstate <= s0;

else

nstate <= s6;

end if;

when s3 =>

if(coin1 = '0') then nstate <= s7;

else

nstate <= s3;

end if;

when s7 =>

if(coin2 = '0') then nstate <= s8;

else

nstate <= s7;

end if;

when s8 =>

if(back = '0') then nstate <= s0;

else

nstate <= s8;

end if;

when s4 =>

if(coin1 = '0') then nstate <= s9;

else

nstate <= s4;

end if;

when s9 =>

if(coin2 = '0') then nstate <= s10;

else

nstate <= s9;

end if;

when s10 =>

if(coin3 = '0') then nstate <= s11;

else

nstate <= s10;

end if;

when s11 =>

if(back = '0') then nstate <= s0;

else

nstate <= s11;

end if;

when s5 =>

if(coin1 = '0') then nstate <= s12;

else

nstate <= s5;

end if;

when s12 =>

if(back = '0') then nstate <= s0;

else

nstate <= s12;

end if;

when others =>

nstate <= s0;

end case;

end process;

next\_state: process(pstate)

begin

case (pstate) is

when s0 =>

dispensor <= "0000";

display1 <= "11000000";

when s1 =>

dispensor <= "0000";

display1 <= "11000000";

when s2 =>

dispensor <= "0000";

display1 <= "11000000";

when s3 =>

dispensor <= "0000";

display1 <= "11000000";

when s4 =>

dispensor <= "0000";

display1 <= "11000000";

when s5 =>

dispensor <= "0000";

display1 <= "11000000";

when s7 =>

dispensor <= "0000";

display1 <= "11111001";

when s9 =>

dispensor <= "0000";

display1 <= "11111001";

when s10 =>

dispensor <= "0000";

display1 <= "10100100";

when s6 =>

dispensor <= "0001";

display1 <= "11111001";

when s8 =>

dispensor <= "0010";

display1 <= "10100100";

when s11 =>

dispensor <= "0100";

display1 <= "10110000";

when s12 =>

dispensor <= "1000";

display1 <= "11111001";

when others =>

dispensor <= "0000";

display1 <= "11000000";

end case;

end process;

end Behavioral;

**Troubleshooting:**

I had some issue with the single pole switches since they are not in the right order as described on the manual. It took me a long time of trial and error to figure that out.

At the beginning the code did not work , so I have to search on the internet on models to do a controller with many outputs and inputs to make it work.

I wanted to implement 3 displays but due to the small time that I have I was not able to do it.

**Recommendations and/or Conclusions:**

I recommend to future students to do a lot of research on any project that they may do with FPGA, because it is not like programming a normal language, it has its formats that one needs to follow in order to make a successful project. In addition, when programming VHDL one needs to think in terms of physical hardware.

This project helps us understand how programming can help to design hardware without the need to connect any wires or logic circuits.

In conclusion, this was a very challenging process to design because one small mistake it can take hours to debug, and to know the source of the problem. Therefore, writing on VHDL is slower than writing on java or C++, or any other traditional language.