ECEN 429: Introduction to Digital Systems Design Laboratory

North Carolina Agricultural and Technical State University

Department of Electrical and Computer Engineering

Michael Newman (Reporter)

Chad Watson (Lab Partner)

December 05, 2019

Lab #10

Introduction

The purpose of this lab was to design and implement a vending machine in VHDL. For **part 1**, The vending machine was designed with the functionality to dispense gum, candy, or soda. (the item being dispensed is represented by the LEDs , and the item is selected using switches on the Basys 3 board). There is also functionality for the machine to signal to the user that “not enough” money was inserted, based on if they make a selection that costs more than the current amount of money in the machine. There is a refund switch that lights up LEDs to represent the amount being refunded. And finally, there is a reset button that allows the machine to return to the default state.

For **part 2** of the lab, we added counters to the implementation, as well as the functionality for the user to receive change back after making a purchase. The three counters added were a money counter (keeps track of how much money has been entered), the gum counter (keeps track of how much gum has been dispensed), the candy counter (tracks how much candy has been dispensed) and the soda counter (tracks how much soda has been dispensed). The money counter is displayed on the Leds, while the 3 “item” counters are displayed on the seven segment display, based on which item is currently selected.

Background, Design Solution, and Results

Part 1:

For part one of this lab, we implemented the vending machine with ability to dispense gum, candy, or soda and show on the LEDs which item is being dispensed. If the user selects an item, then they can enter a nickel or a dime (represented by buttons on the board) and the machine will either output the item selected, or if not enough money has been entered for the item, the machine will display the insufficient funds LED. Figure 1.1 below shows the pin mapping for this design.

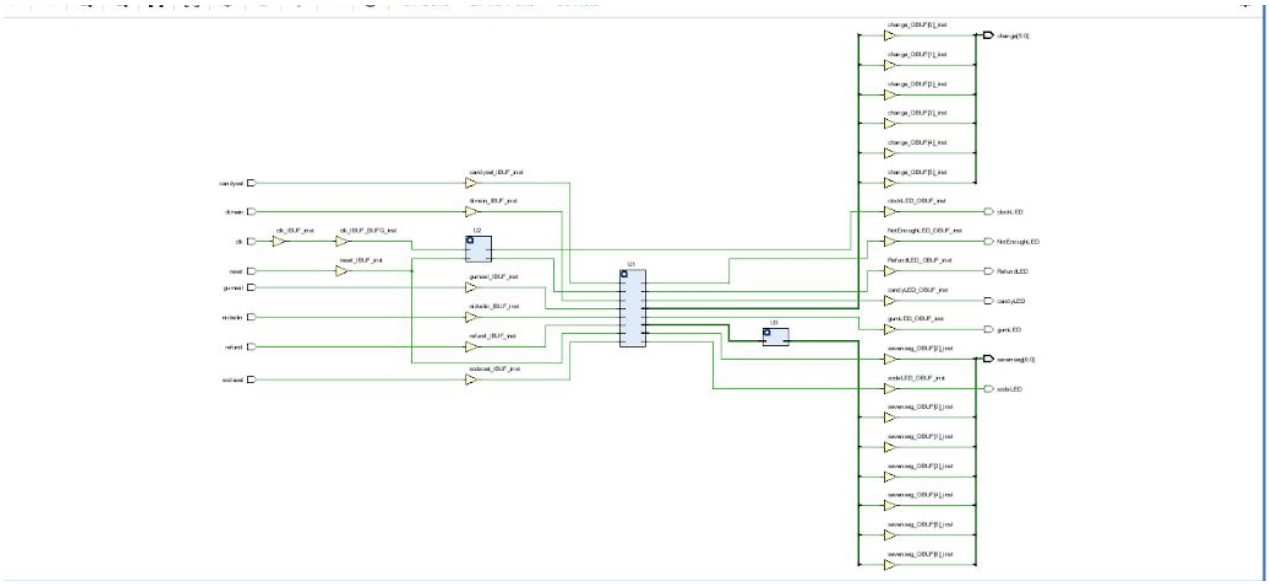

Figure 1.1 This is a schematic of the Vending Machine in part 1

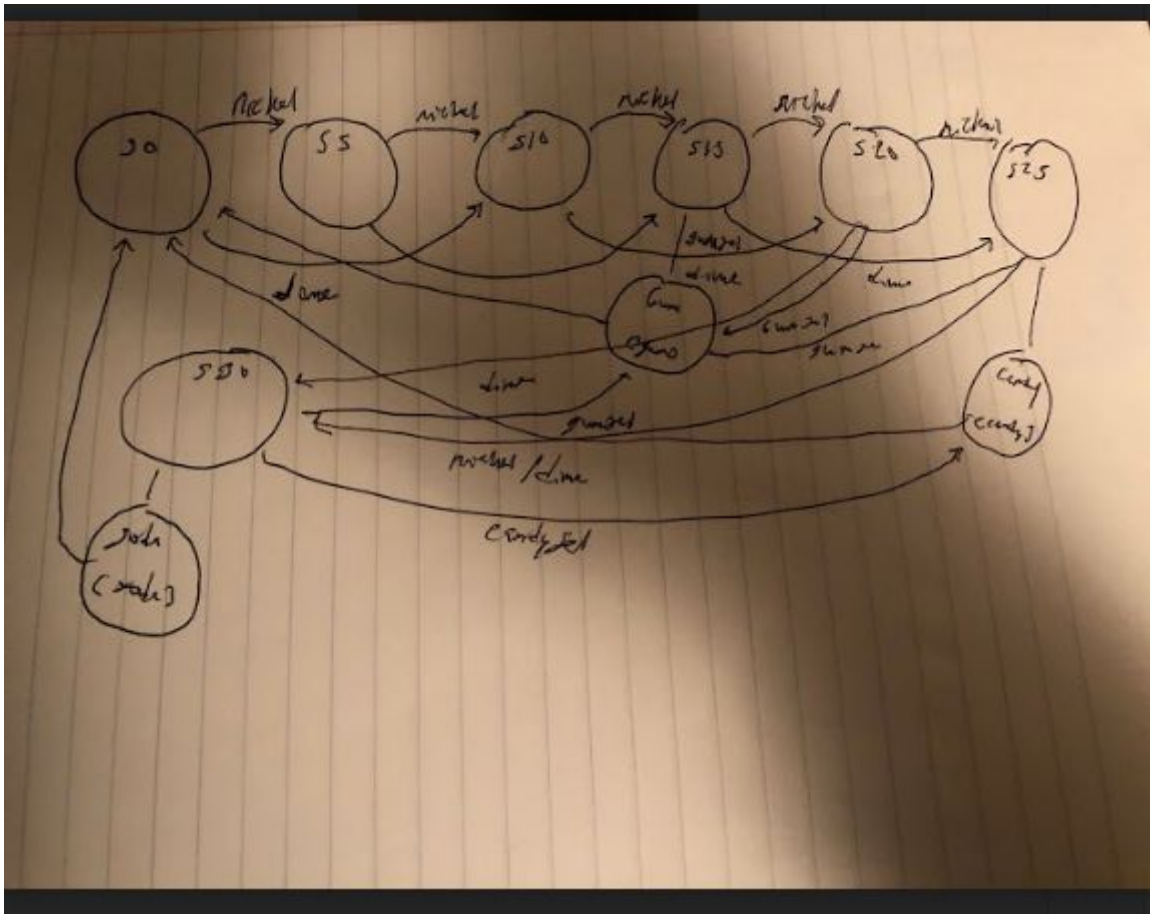

Figure 1.2 This is the FSM for the vending machine

FPGA Pin Assignment for Vending Machine

input/output	Pin Assignment
clk:input	W5

gumsel: input	R2
candysel:input	T1
sodasel:input	U1
refund:input	T2
nickelin:input	W19
dimein:input	T17
NotEnoughLED:output	E19
clockLED:output	U16
GumLED:output	L1
CandyLED:output	P1
SodaLED:output	N3
RefundLED:output	U19
change(5):output	W3
change(4);output	V3
change(3):output	V13
change(2):output	V14
change(1):output	U14
change(0):output	U15
outTimer(6):output	W7
outTimer(5):output	V5
outTimer(4):output	U5

outTimer(3):output	V8
outTimer(2):output	U8
outTimer(1):output	W6
outTimer(0):output	U7
Reset: input	T3

Table 1.1: Displays the input/output and the corresponding FPGA pin assignment for the Vending Machine

Results on FPGA board

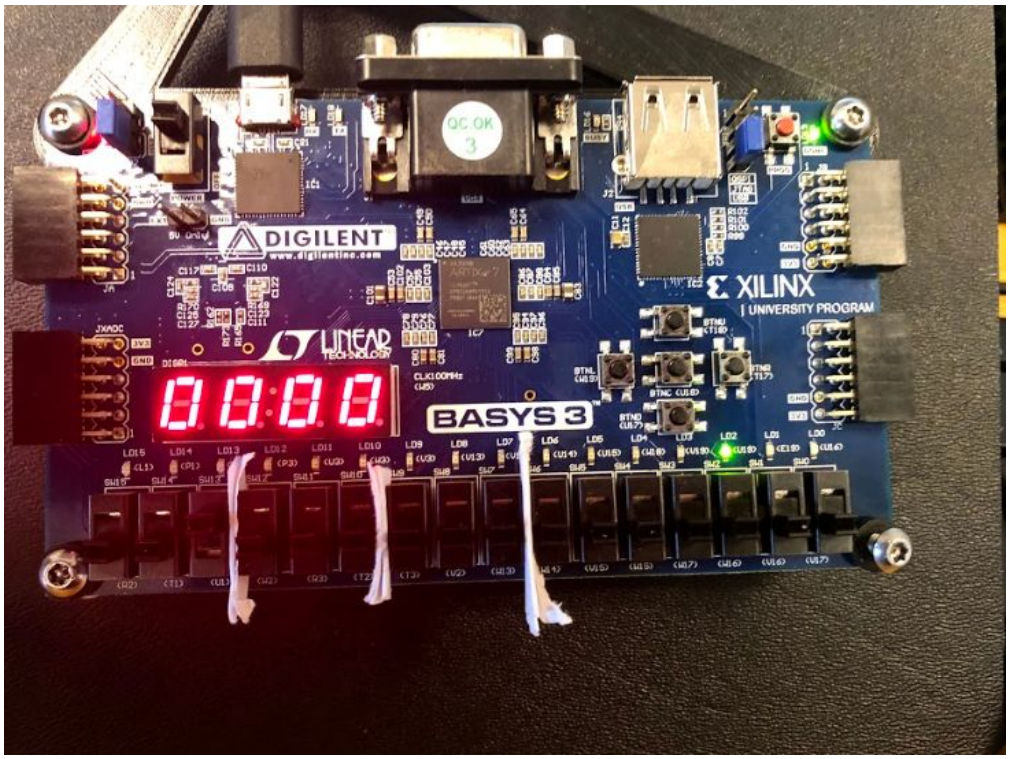

Figure 1.3 This is a picture of the vending machine where soda was selected in state 0 or S0 so the NotEnoughLED comes on

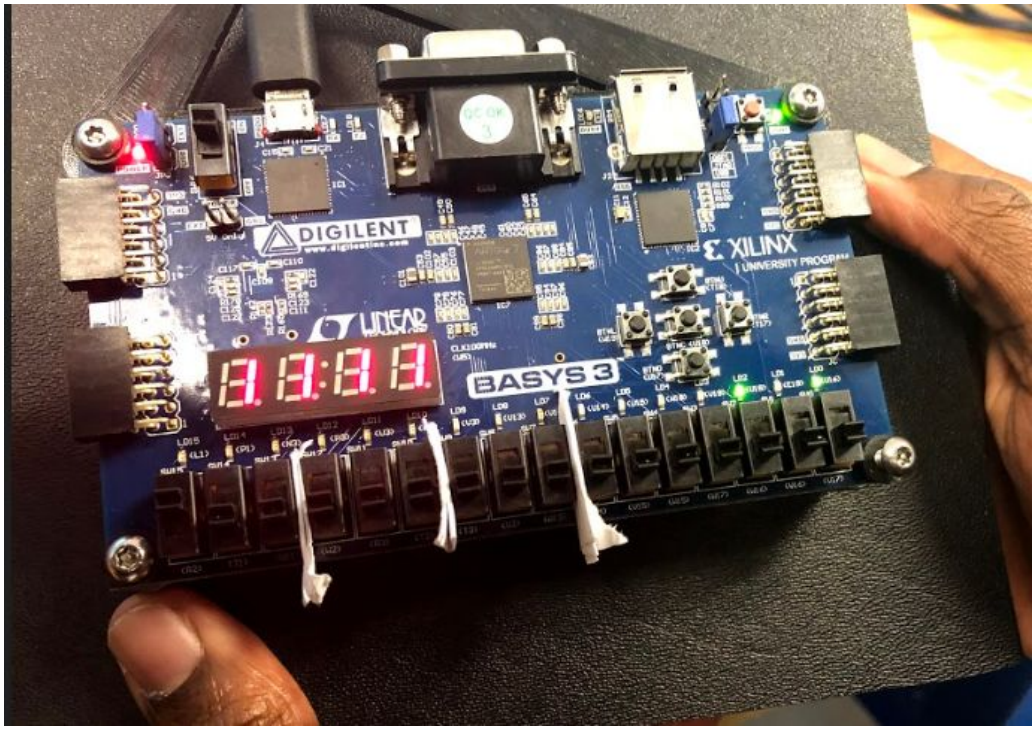

Figure 1.4 This is a picture of the vending machine where gum was selected in state 1 or S5 so the NotEnoughLED comes on

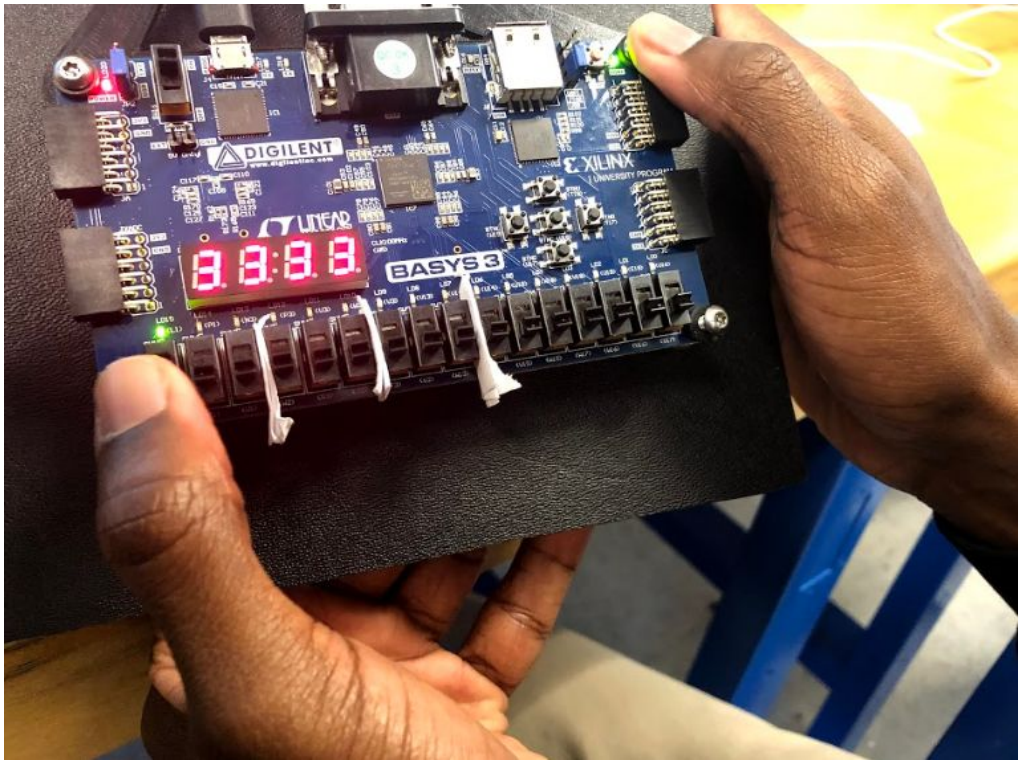

Figure 1.5 This is a picture of the vending machine where gum was selected in state 3 or S15 so gum was dispensed by the gumLED coming on

Part 2:

For part 2 of this lab, we added more functionality to the vending machine, allowing it to keep track of how many times an item has been dispensed (using a separate counter for each item), and also allowing the user to receive change (up to 15 cents) for a selection. There is also another money counter that keeps track of how much money has entered the system. As in part 1, there is an insufficient funds light, and a refund switch that allows the user to refund their money. The reset switch is also in this implementation, allowing the user to return to the default state. Figure 2.1 below shows the schematic for this design.

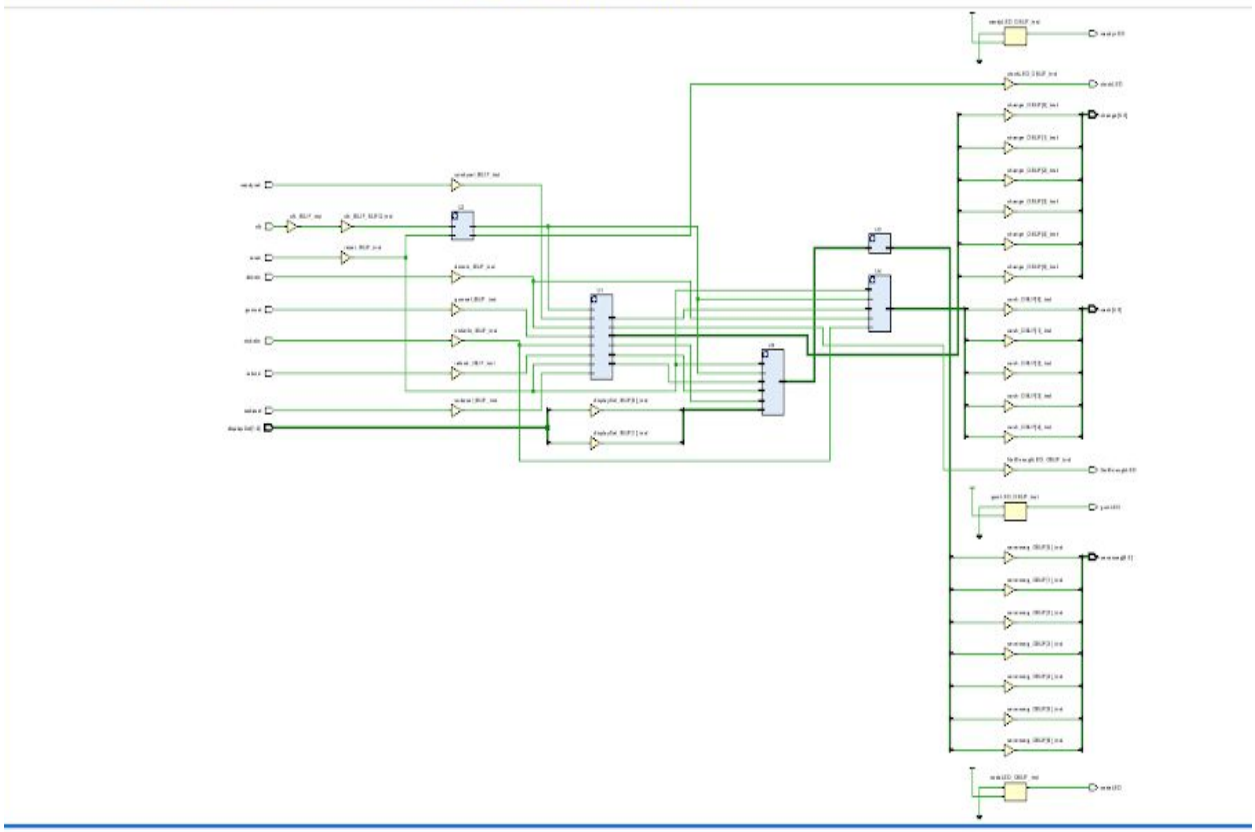

Figure 2.1 shows the schematic for the design

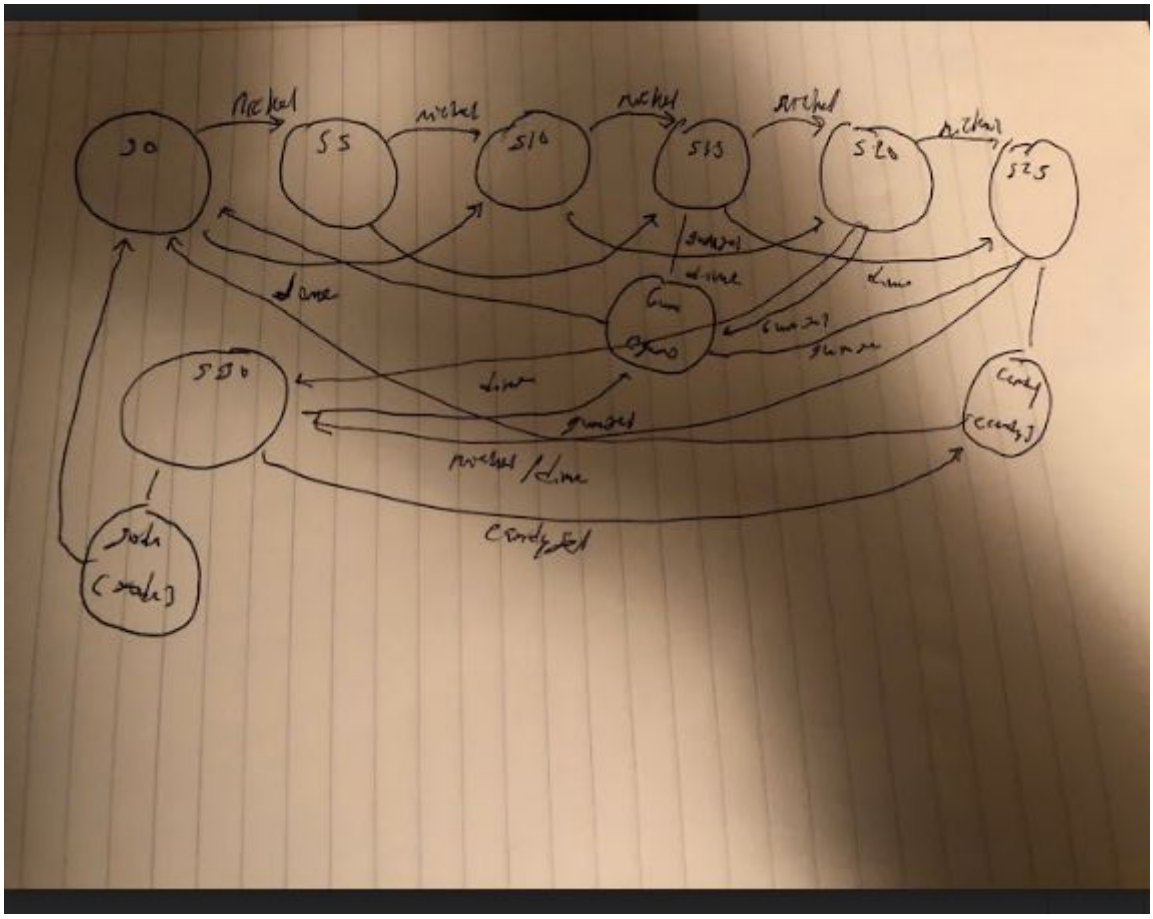

Figure 2.2 This is FSM for the vending machine

FPGA Pin Assignment for Vending Machine

input/output	Pin Assignment
clk:input	W5
gumsel: input	R2
candysel:input	T1
sodasel:input	U1
refund:input	T2
nickelin:input	W19

dimein:input	T17
NotEnoughLED:output	E19
clockLED:output	U16
RefundLED:output	U19
change(5):output	W3
change(4);output	V3
change(3):output	V13
change(2):output	V14
change(1):output	U14
change(0):output	U15
cash(4):output	L1
cash(3):output	P1
cash(2):output	N3
cash(1):output	P3
cash(0):output	U3
sevenseg(6):output	W7
sevenseg(5):output	V5
sevenseg(4):output	U5
sevenseg(3):output	V8
sevenseg(2):output	U8
sevenseg(1):output	W6

sevenseg(0):output	U7
Reset: input	T3

Table 2.1: Displays the input/output and the corresponding FPGA pin assignment for the Vending Machine

Results on FPGA board

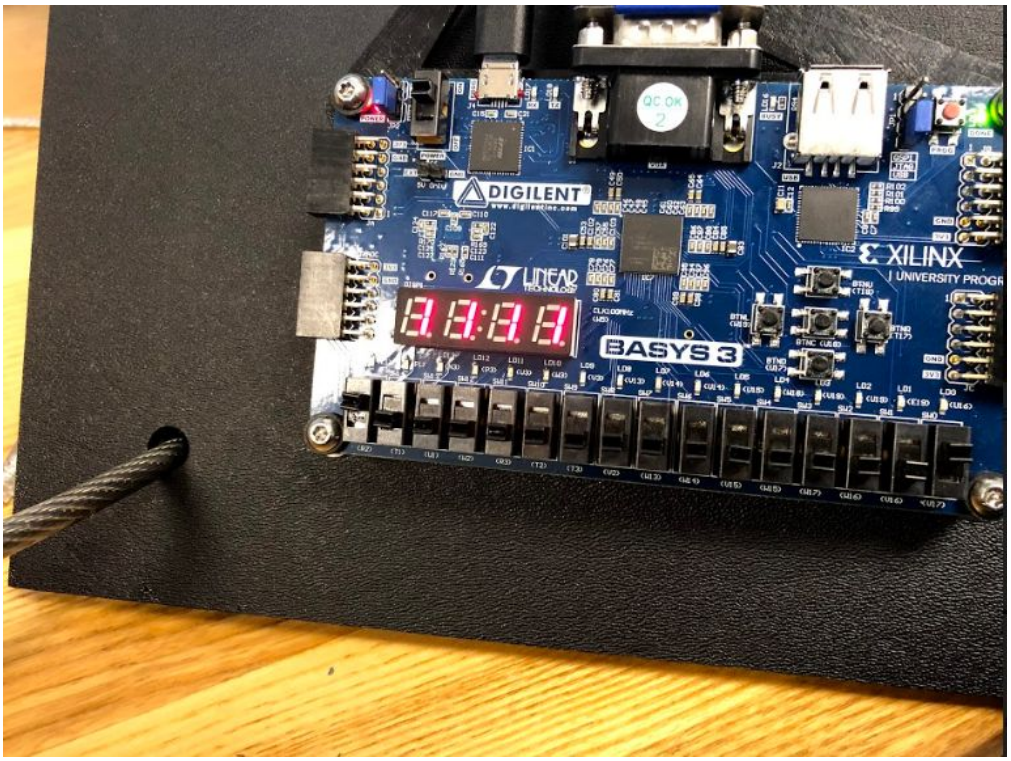

Figure 2.3 This is a picture of the vending machine showing the counter for candy be dispensed one time

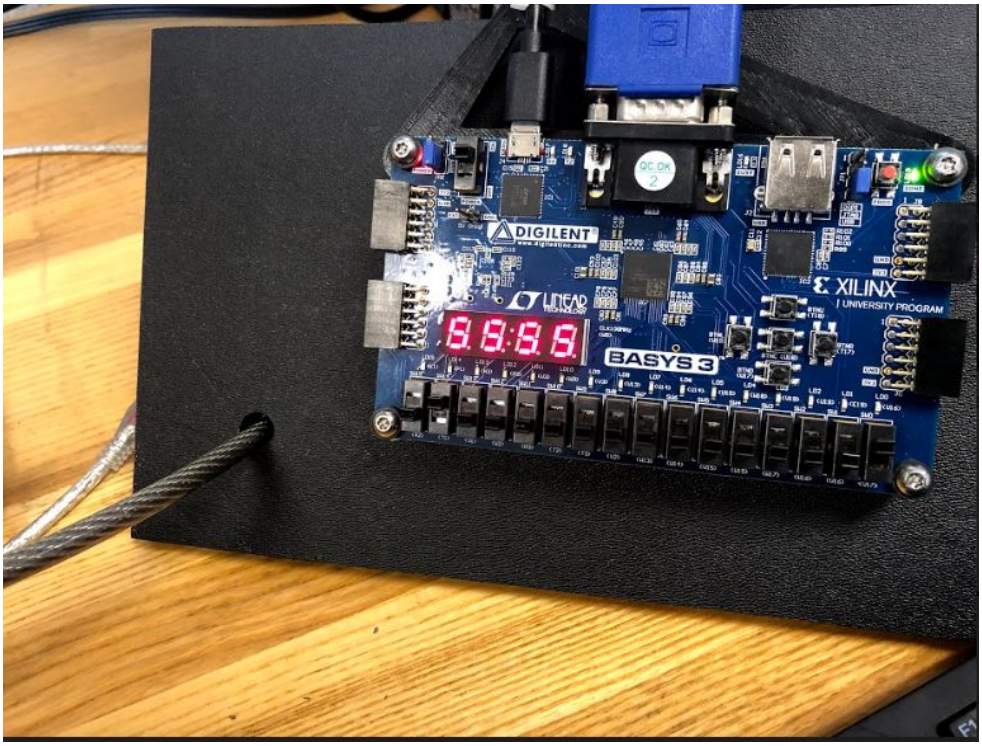

Figure 2.4 This is a picture of the vending machine with the select for the gum counter active showing gum was dispensed 5 times

Conclusion:

In conclusion, this lab was an interesting and engaging lab. We ran into some challenges early on in part one, by accidentally mapping our clock to the wrong output. This caused our implementation to be buggy and not move to the correct states. After we realized this error, however, we were well on our way to implementing the vending machine to meet the constraints of the assignment. On part two, it took quite a bit of thought to figure out how to code the counters and also use the remaining LEDs on the board in order to cooperate with the instructions of the assignment. In the end we were able to develop a working implementation, and we learned a lot about using multiple counters in a state machine design, as well as touched up on the modular programming approach to VHDL.

VHDL for parts of lab

Part 1

TOP LEVEL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity Top is port (clk, reset, nickelin,dimein: in std_logic;
                refund, gumsel,candysel,sodasel:in std_logic;
                gumLED,candyLED, sodaLED, NotEnoughLED,RefundLED, clockLED: out
std_logic;
                change: out std_logic_vector(5 downto 0);
                sevenseg: out std_logic_vector(6 downto 0));
  end Top;

architecture Behavioral of Top is
signal clockOut:std_logic;
signal SevenSegOut : std_logic_vector( 3 downto 0);
component Clockdivider is port(clk : in std_logic;
         reset : in std_logic;
          SlowClock, led0 : out std_logic);
          end component;

component VM is port(clk, reset,nickel,dime: in std_logic;
          refund, gumsel, candysel, sodasel: in std_logic;
          gumLED,candyLED, sodaLED, NotEnoughLED,RefundLED: out std_logic;
          change: out std_logic_vector(5 downto 0);
          showState: out std_logic_vector(3 downto 0));
      end component;

    component display is port(inp: in std_logic_vector (3 downto 0);
                output: out std_logic_vector(6 downto 0) );
                end component;
begin
    U1:VM port map(clk=>clockOut, reset=>reset, nickel=> nickelin, dime=> dimein,
refund=>refund, gumsel=>gumsel,candysel=>candysel, sodasel=>sodasel,
                  gumLED=> gumLED, candyLED=> candyLED,sodaLED=>sodaLED,
NotEnoughLED=> NotEnoughLED, RefundLED=>RefundLED,
change=>change,showState=>SevenSegOut);
    U2: Clockdivider port map(clk=>clk,reset=>reset,SlowClock=> clockOut, led0=>clockLED);
    U3: display port map (inp => SevenSegOut, output =>sevenseg);
end Behavioral;
```

Clock Divider

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.All;
entity Clockdivider is
     port(clk : in std_logic;
          reset : in std_logic;
```

```
        SlowClock, led0 : out std_logic);
end Clockdivider;

architecture behavioral of Clockdivider is
--signal for clock
signal slowClock_Sig:std_logic;
begin
    process

        variable cnt :   std_logic_vector(26 downto 0):= "00000000000000000000000000";
   --Counter. When this variable reches MSB approximately one second (1 Hz) will have passed
        begin

                wait until ((clk'EVENT) AND (clk = '1')); --wait until clock is high

                if (reset = '1') then
                --reset the counter
                cnt := "00000000000000000000000000";
            else
           --start counting by incrementing by 1
           cnt := std_logic_vector (unsigned (cnt) + 1);
            end if;


        SlowClock <= cnt(26);  --SlowClock = 1 Hz
        slowClock_sig <= cnt(26);--same as slow clock

        --Display the clock info to the LED
        if (SlowClock_sig = '1') then
                led0 <= '1';
        else
                led0 <= '0';
        end if;
 end process;
end behavioral;
```

VM FSM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity VM is
 Port (clk, reset,nickel,dime: in std_logic;
       refund, gumsel, candysel, sodasel: in std_logic;
       gumLED,candyLED, sodaLED, NotEnoughLED,RefundLED: out std_logic;
       change: out std_logic_vector(5 downto 0);
       showState: out std_logic_vector(3 downto 0));
end VM;

architecture Behavioral of VM is
 type state_type is (S0,S5,S10,S15, S20, S25,S30);
signal CS, NS: state_type;
begin
    process(clk,reset)
   begin
    if (reset='1')then
    CS <= S0;
    elsif (clk'event and clk='1')then
    CS<= NS;
    end if;
    end process;

    process(CS, NS, nickel, dime, gumsel,candysel, sodasel)
    begin
--   NotEnoughLED <='0';
--   gumLED <='0';
--    sodaLED <= '0';
--    candyLED <='0';
--   change<="000000";
--   RefundLED <= '0';

case CS is

when S0=>
   NotEnoughLED <='0';
gumLED <='0';
 sodaLED <= '0';
 candyLED <='0';
change<="000000";
RefundLED <= '0';
 showState<="0000";


     if(nickel ='1')then
```

```
      NS <=S5;
     elsif (dime ='1')then
      NS <= S10;
    elsif((nickel ='0' and dime='0'))then
    NS <= CS;
    end if;

    if (refund ='1') then
      RefundLED<= '1';
      change<="000000";
      NS<=S0;
      end if;

  if(gumsel='1' or candysel='1' or sodasel='1')then
  NotEnoughLED <='1';
    end if;


when S5=>
--gumLED<='1'; --delete this L1
--   sodaLED<='0'; --delete this n3
  showState<="0001";
--    candyLED <='0';
     change<="000000";

        if (nickel='1') then
        NS <=S10;
        elsif(dime = '1') then
        NS<=S15;
      else
      NS<=CS;
      end if;

  if(gumsel ='1' or candysel='1' or sodasel='1') then
      NotEnoughLED <='1';
      end if;


    if (refund ='1')then
    RefundLED<= '1';
    change<="000001";
    NS<=S0;
    end if;
```

```
when S10=>
--  candyLED<='1';--delete p1
--   gumLED<='0';
--   sodaLED<='0';
    change<="000000";
   showState<="0010";
   if (nickel='1') then
          NS <=S15;
          elsif(dime = '1') then
          NS<=S20;
         else
         NS<=CS;
         end if;


   if(gumsel ='1' or candysel='1' or sodasel='1') then
        NotEnoughLED <='1';
        NS <= S10;
  end if;

   if(refund ='1')then
   RefundLED<= '1';
   change<="000010";
   NS<=S0;
   end if;

 when S15=>
 change<="000000";
 showState<="0011";
 if (nickel='1') then
           NS <=S20;
           elsif(dime = '1') then
           NS<=S25;
          else
          NS<=CS;
          end if;


    if(candysel='1' or sodasel='1') then
          NotEnoughLED <='1';
```

```
        NS <= CS;
   end if;

   if(gumsel = '1') then
   gumLED <= '1';
   NS <= S0;
   end if;

    if(refund ='1')then
    RefundLED<= '1';
    change<="000100";
    NS<=S0;
    end if;


when S20=>
 change<="000000";
 showState<="0100";
 if (nickel='1') then
          NS <=S25;
          elsif(dime = '1') then
          NS<=S30;
         else
         NS<=CS;
         end if;


   if(candysel='1' or sodasel='1') then
        NotEnoughLED <='1';
        NS <= CS;
  end if;

  if(gumsel = '1') then
  gumLED <= '1';
  NS <= S0;
  end if;

   if(refund ='1')then
   RefundLED<= '1';
   change<="001000";
   NS<= S0;
   end if;
```

```
when S25=>
 change<="000000";
 showState<="0101";
if (nickel='1') then
          NS <=S30;
          elsif(dime = '1') then
          NS<=S30;
        else
        NS<=CS;
        end if;


  if(sodasel='1') then
        NotEnoughLED <='1';
        NS <= CS;
 end if;

 if(gumsel = '1') then
 gumLED <= '1';
 NS <= S0;
 end if;

 if(candysel = '1') then
 candyLED <= '1';
 NS <= S0;
 end if;

  if(refund ='1')then
  RefundLED<= '1';
  change<="010000";
  NS <= S0;
  end if;


 when S30=>
 change<="000000";
 showState<="0110";
if (nickel='1') then
          NS <=S30;
          elsif(dime = '1') then
          NS<=S30;
        else
```

```
        NS<=CS;
        end if;

 if(gumsel = '1') then
 gumLED <= '1';
 NS <= S0;
 end if;

 if(candysel = '1') then
 candyLED <= '1';
 NS <= S0;
 end if;

 if(sodasel = '1') then
  sodaLED <= '1';
  NS <= S0;
  end if;

  if(refund ='1')then
  RefundLED<= '1';
  change<="100000";
  NS <= S0;
  end if;

    end case;
    end process;

end Behavioral;
```

Constraints File

```
set_property IOSTANDARD LVCMOS33 [get_ports {change[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[0]}]

set_property PACKAGE_PIN W18 [get_ports {change[5]}]
set_property PACKAGE_PIN U15 [get_ports {change[4]}]
```

```
set_property PACKAGE_PIN U14 [get_ports {change[3]}]
set_property PACKAGE_PIN V14 [get_ports {change[2]}]
set_property PACKAGE_PIN V13 [get_ports {change[1]}]
set_property PACKAGE_PIN V3 [get_ports {change[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports candyLED]
set_property IOSTANDARD LVCMOS33 [get_ports candysel]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clockLED]
set_property IOSTANDARD LVCMOS33 [get_ports dimein]
set_property IOSTANDARD LVCMOS33 [get_ports gumLED]
set_property IOSTANDARD LVCMOS33 [get_ports gumsel]
set_property IOSTANDARD LVCMOS33 [get_ports nickelin]
set_property IOSTANDARD LVCMOS33 [get_ports NotEnoughLED]
set_property IOSTANDARD LVCMOS33 [get_ports refund]
set_property IOSTANDARD LVCMOS33 [get_ports RefundLED]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports sodaLED]
set_property IOSTANDARD LVCMOS33 [get_ports sodasel]



set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[0]}]
set_property PACKAGE_PIN W7 [get_ports {sevenseg[6]}]
set_property PACKAGE_PIN U7 [get_ports {sevenseg[0]}]
set_property PACKAGE_PIN W6 [get_ports {sevenseg[1]}]
set_property PACKAGE_PIN U8 [get_ports {sevenseg[2]}]
set_property PACKAGE_PIN V8 [get_ports {sevenseg[3]}]
set_property PACKAGE_PIN U5 [get_ports {sevenseg[5]}]
set_property PACKAGE_PIN V5 [get_ports {sevenseg[4]}]
```

VHDL for parts of lab

Part 2

TOP LEVEL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity Top is port (clk, reset, nickelin,dimein: in std_logic;
                    refund, gumsel,candysel,sodasel:in std_logic;
                    gumLED,candyLED, sodaLED, NotEnoughLED, clockLED: out std_logic;
                    displaySel:in std_logic_vector(1 downto 0);
                    change: out std_logic_vector(5 downto 0);
                    cash: out std_logic_vector(4 downto 0);
                    sevenseg: out std_logic_vector(6 downto 0));
  end Top;

architecture Behavioral of Top is
signal clockOut,nickleSig,dimeSig,gumout,sodaout,candyout :std_logic;
signal total : std_logic_vector( 3 downto 0);
component Clockdivider is port(clk : in std_logic;
          reset : in std_logic;
           SlowClock, led0 : out std_logic);
           end component;

component VM is port(clk, reset,nickel,dime: in std_logic;
           refund, gumsel, candysel, sodasel: in std_logic;
           gumLED,candyLED, sodaLED, NotEnoughLED,nickleCount,dimeCount: out std_logic;
           change: out std_logic_vector(5 downto 0));
      end component;

    component display is port(inp: in std_logic_vector (3 downto 0);
                output: out std_logic_vector(6 downto 0) );
                end component;
    component MoneyCounter is port(
                clock,nickin, dimin,gumout,candyout,sodaout: in std_logic;
                R: in std_logic;
                moneycount: out std_logic_vector(4 downto 0));
            end component;
    component ItemCounter is port(clock,gumin,sodain,candyin: in std_logic;
            sel: in std_logic_vector(1 downto 0);
            R: in std_logic;
            total: out std_logic_vector(3 downto 0));
            end component;
```

```
begin
   U1:VM port map(clk=>clockOut, reset=>reset, nickel=> nickelin, dime=> dimein,
refund=>refund, gumsel=>gumsel,candysel=>candysel, sodasel=>sodasel,
                  gumLED=> gumout, candyLED=> candyout,sodaLED=>sodaout,
NotEnoughLED=> NotEnoughLED, change=>change, nickleCount=>nickleSig,
dimeCount=>dimeSig);
   U2: Clockdivider port map(clk=>clk,reset=>reset,SlowClock=> clockOut, led0=>clockLED);
   U3: display port map (inp => total, output =>sevenseg);
   U4: MoneyCounter port map(clock=>clockOut, R=>reset, nickin=>nickleSig,
dimin=>dimeSig,gumout=>gumout,sodaout=>sodaout,candyout=>candyout,moneycount=>cash
);
   U5: ItemCounter port map(clock=>clockOut,R=>reset,gumin=> gumout, sodain=>sodaout,
candyin=> candyout,total=>total, sel=>displaySel);

end Behavioral;
```

Item Counter

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

```
use ieee.std_logic_unsigned.all; -- need this to add std_logic_vectors

use IEEE.NUMERIC_STD.ALL;

entity ItemCounter is port (
clock,gumin,sodain,candyin: in std_logic;
sel: in std_logic_vector(1 downto 0);
R: in std_logic;
total: out std_logic_vector(3 downto 0));
end ItemCounter;
architecture behavioral of ItemCounter is
signal tmpG,tmpS,tmpC: std_logic_vector(3 downto 0);
begin
process (clock, R)
begin
if (R = '1') then
tmpG<="0000"; --reset the counter
elsif (rising_edge(clock))then
    if (gumin='1')then
        tmpG <= tmpG+ "0001"; --start counter for gum
   elsif (sodain='1')then
            tmpS <= tmpS+ "0001"; --start counter for soda

    elsif (candyin='1')then
            tmpC <= tmpC+ "0001"; --start counter for candy
end if;
end if;
if (sel="01")then
total<=tmpG;
elsif (sel="10")then
total<=tmpC;
elsif (sel="11")then
total<=tmpS;
end if;
end process;

end behavioral;
```

Money Counter

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

```
use ieee.std_logic_unsigned.all; -- need this to add std_logic_vectors

use IEEE.NUMERIC_STD.ALL;

entity MoneyCounter is port (
clock,nickin, dimin,gumout,candyout,sodaout: in std_logic;
R: in std_logic;
moneycount: out std_logic_vector(4 downto 0));
end MoneyCounter;
architecture behavioral of MoneyCounter is
signal tmp: std_logic_vector(4 downto 0);
begin
process (clock, R)
begin
if (R = '1') then
tmp<="00000"; --reset the counter
elsif (rising_edge(clock))then
   if (nickin='1')then
      tmp <= tmp+"0101"; --add 5
   elsif(dimin='1')then
         tmp <= tmp +"1010";-- add 10
   elsif(gumout ='1' or candyout='1' or sodaout='1')then
     tmp<="00000";
end if;
end if;

end process;
moneycount<= tmp;
end behavioral;
```

VM FSM VHDL

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;



entity VM is
 Port (clk, reset,nickel,dime: in std_logic;
       refund, gumsel, candysel, sodasel: in std_logic;
       gumLED,candyLED, sodaLED, NotEnoughLED,nickleCount,dimeCount: out std_logic;
       change: out std_logic_vector(5 downto 0));
end VM;

architecture Behavioral of VM is
 type state_type is (S0,S5,S10,S15, S20, S25,S30);
signal CS, NS: state_type;
begin
    process(clk,reset)
   begin
    if (reset='1')then
    CS <= S0;
    elsif (clk'event and clk='1')then
    CS<= NS;
    end if;
    end process;

    process(CS, NS, nickel, dime, gumsel,candysel, sodasel)
    begin
--   NotEnoughLED <='0';
--   gumLED <='0';
--    sodaLED <= '0';
--    candyLED <='0';
--   change<="000000";
--   RefundLED <= '0';
    nickleCount<=nickel;
    dimeCount<= dime;

case CS is

when S0=>
   NotEnoughLED <='0';
gumLED <='0';
 sodaLED <= '0';
 candyLED <='0';
change<="000000";
```

```
--RefundLED <= '0';



     if(nickel ='1')then
       NS <=S5;
      elsif (dime ='1')then
       NS <= S10;
     elsif((nickel ='0' and dime='0'))then
     NS <= CS;
     end if;

     if (refund ='1') then
      -- RefundLED<= '1';
       change<="000000";
       NS<=S0;
       end if;

   if(gumsel='1' or candysel='1' or sodasel='1')then
   NotEnoughLED <='1';
     end if;


when S5=>
--gumLED<='1'; --delete this L1
--   sodaLED<='0'; --delete this n3

--    candyLED <='0';
     change<="000000";

         if (nickel='1') then
         NS <=S10;
         elsif(dime = '1') then
         NS<=S15;
       else
       NS<=CS;
       end if;

   if(gumsel ='1' or candysel='1' or sodasel='1') then
      NotEnoughLED <='1';
      end if;
```

```
    if (refund ='1')then
    --RefundLED<= '1';
    change<="000001";
    NS<=S0;
    end if;



when S10=>
--  candyLED<='1';--delete p1
--   gumLED<='0';
--   sodaLED<='0';
    change<="000000";

   if (nickel='1') then
          NS <=S15;
          elsif(dime = '1') then
          NS<=S20;
         else
         NS<=CS;
         end if;


   if(gumsel ='1' or candysel='1' or sodasel='1') then
        NotEnoughLED <='1';
        NS <= S10;
  end if;

   if(refund ='1')then
  -- RefundLED<= '1';
   change<="000010";
   NS<=S0;
   end if;

 when S15=>
 change<="000000";

 if (nickel='1') then
           NS <=S20;
           elsif(dime = '1') then
           NS<=S25;
          else
          NS<=CS;
```

```
        end if;


    if(candysel='1' or sodasel='1') then
        NotEnoughLED <='1';
        NS <= CS;
   end if;

   if(gumsel = '1') then
   gumLED <= '1';
   NS <= S0;
   end if;

    if(refund ='1')then
   -- RefundLED<= '1';
    change<="000100";
    NS<=S0;
    end if;


when S20=>
 change<="000000";

 if (nickel='1') then
          NS <=S25;
          elsif(dime = '1') then
          NS<=S30;
        else
        NS<=CS;
        end if;


   if(candysel='1' or sodasel='1') then
        NotEnoughLED <='1';
        NS <= CS;
  end if;

  if(gumsel = '1') then
  gumLED <= '1';
 change<="000001";
 NS <= S0;
  end if;
```

```
  if(refund ='1')then
 -- RefundLED<= '1';
  change<="001000";
  NS<= S0;
  end if;


when S25=>
 change<="000000";

if (nickel='1') then
         NS <=S30;
         elsif(dime = '1') then
         NS<=S30;
        else
        NS<=CS;
        end if;


 if(sodasel='1') then
       NotEnoughLED <='1';
       NS <= CS;
 end if;

 if(gumsel = '1') then
 gumLED <= '1';
 change <="000010";
 NS <= S0;
 end if;

 if(candysel = '1') then
 candyLED <= '1';
 NS <= S0;
 end if;

 if(refund ='1')then
 --RefundLED<= '1';
 change<="010000";
 NS <= S0;
 end if;


 when S30=>
```

```
 change<="000000";

if (nickel='1') then
          NS <=S30;
          elsif(dime = '1') then
          NS<=S30;
         else
         NS<=CS;
         end if;

 if(gumsel = '1') then
 gumLED <= '1';
 change <="000100";
 NS <= S0;
 end if;

 if(candysel = '1') then
 candyLED <= '1';
 change <="000001";
 NS <= S0;
 end if;

 if(sodasel = '1') then
  sodaLED <= '1';
  NS <= S0;
  end if;

  if(refund ='1')then
  --RefundLED<= '1';
  change<="100000";
  NS <= S0;
  end if;

    end case;
    end process;

end Behavioral;
```

Display VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity display is
        Port (inp: in std_logic_vector (3 downto 0);
           output: out std_logic_vector(6 downto 0) );
end display;

architecture Behavioral of display is

begin
process(inp)
begin
case inp is
when "0000" => output <= "0000001";--0
when "0001" => output <= "1111001";  -- 1
when "0010" => output <= "0100100";--2
when "0011" => output <= "0110000"; --3
when "0100" => output <= "1011000"; --4

                    --  afedcbg
when "0101" => output <= "0010010"; --5
when "0110" => output <= "0000010"; --6
when "0111" => output <= "0111001"; --7
when "1000" => output <= "0000000"; --8
when "1001" => output <= "0010000"; --9
when "1010" => output <= "0001000"; --A
when "1011" => output <= "1100000"; --B
when "1100" => output <= "0000111"; --C
when "1101" => output <= "1100000"; --D
when "1110" => output <= "0000110"; --E
when "1111" => output <= "0001110"; --F
end case;
end process;
end Behavioral;
```

Constraints File

```
set_property IOSTANDARD LVCMOS33 [get_ports {cash[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {cash[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {cash[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {cash[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {cash[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {change[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {displaySel[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {displaySel[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sevenseg[0]}]
set_property PACKAGE_PIN L1 [get_ports {cash[4]}]
set_property PACKAGE_PIN P1 [get_ports {cash[3]}]
set_property PACKAGE_PIN N3 [get_ports {cash[2]}]
set_property PACKAGE_PIN P3 [get_ports {cash[1]}]
set_property PACKAGE_PIN U3 [get_ports {cash[0]}]
set_property PACKAGE_PIN W3 [get_ports {change[5]}]
```

```
set_property PACKAGE_PIN V3 [get_ports {change[4]}]
set_property PACKAGE_PIN V13 [get_ports {change[3]}]
set_property PACKAGE_PIN V14 [get_ports {change[2]}]
set_property PACKAGE_PIN U14 [get_ports {change[1]}]
set_property PACKAGE_PIN U15 [get_ports {change[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports NotEnoughLED]
set_property IOSTANDARD LVCMOS33 [get_ports nickelin]
set_property IOSTANDARD LVCMOS33 [get_ports gumsel]
set_property IOSTANDARD LVCMOS33 [get_ports gumLED]
set_property IOSTANDARD LVCMOS33 [get_ports dimein]
set_property IOSTANDARD LVCMOS33 [get_ports clockLED]
set_property IOSTANDARD LVCMOS33 [get_ports candyLED]
set_property IOSTANDARD LVCMOS33 [get_ports candysel]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN U16 [get_ports clockLED]
set_property PACKAGE_PIN E19 [get_ports NotEnoughLED]
set_property PACKAGE_PIN W18 [get_ports gumLED]
set_property PACKAGE_PIN V19 [get_ports candyLED]
set_property IOSTANDARD LVCMOS33 [get_ports sodaLED]
set_property PACKAGE_PIN U19 [get_ports sodaLED]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports refund]
set_property IOSTANDARD LVCMOS33 [get_ports sodasel]
set_property PACKAGE_PIN R2 [get_ports {displaySel[1]}]
set_property PACKAGE_PIN T1 [get_ports {displaySel[0]}]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN W19 [get_ports nickelin]
set_property PACKAGE_PIN T17 [get_ports dimein]
set_property PACKAGE_PIN U1 [get_ports gumsel]
set_property PACKAGE_PIN W2 [get_ports candysel]
set_property PACKAGE_PIN R3 [get_ports sodasel]
set_property PACKAGE_PIN T2 [get_ports refund]
set_property PACKAGE_PIN T3 [get_ports reset]
set_property PACKAGE_PIN W7 [get_ports {sevenseg[6]}]
set_property PACKAGE_PIN V5 [get_ports {sevenseg[5]}]
set_property PACKAGE_PIN U5 [get_ports {sevenseg[4]}]
set_property PACKAGE_PIN V8 [get_ports {sevenseg[3]}]
set_property PACKAGE_PIN U8 [get_ports {sevenseg[2]}]
set_property PACKAGE_PIN W6 [get_ports {sevenseg[1]}]
set_property PACKAGE_PIN U7 [get_ports {sevenseg[0]}]
```