VENDING MACHINE TECHNICAL DOCUMENTATION

Team H

15.06.23

1 Team members

Maxim Emile Speczyk

Muhammad Umer Bin Yaqoob

Stephanie Chinenye Okosa

2 Concept description

In this project, we aim to design a hardware system for a vending machine using Field-Programmable Gate Arrays (FPGA), as well as designing our own PCB to its specific needs.
The vending machine's main applications are to dispense an item, accept coins, as well as calculate and dispense the change. For this we use switches to select the item and coin input. Also, the 7-segment display shows the cost of an item, refund price and the insufficient amount.

Figure 1 is the state machine of the vending machine. In the state "Idle", the system waits for input of the item that is requested. After the item is requested, the money needs to be accepted, which happens in the state "Accepting". After enough money is detected, the item gets dispensed and the state changes to "Dispensing". Then the system changes to the state "Refund" which includes calculating and giving out the change back to the customer.

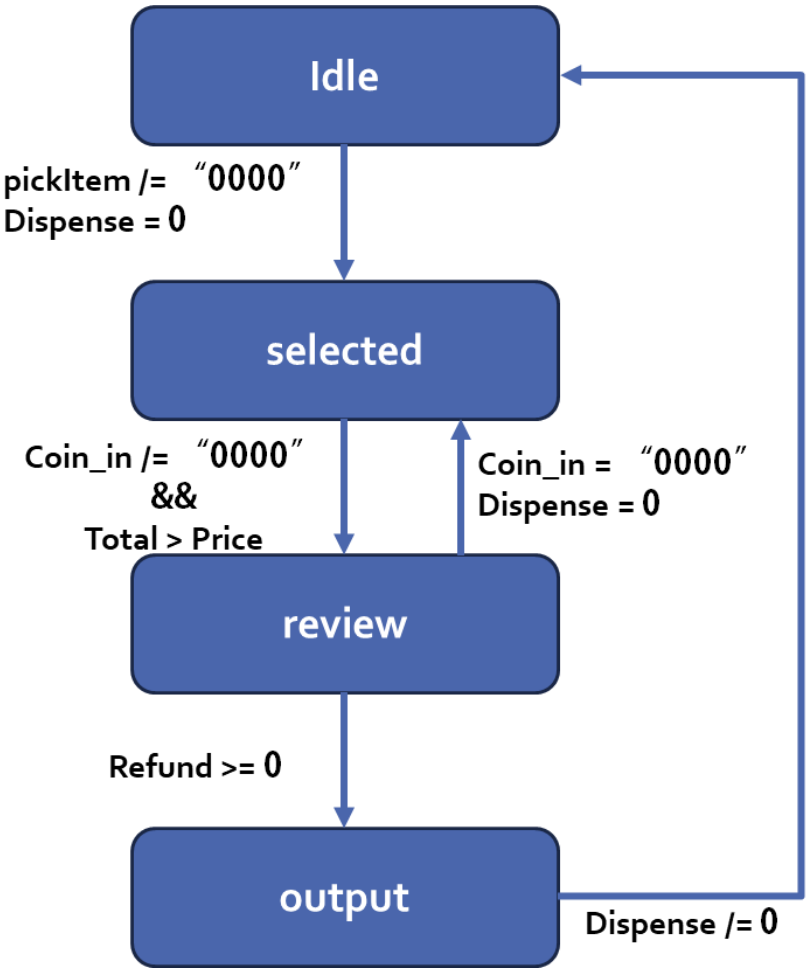


Figure 1: Vending Machine FSM

Below is the block diagram which illustrates the components of the vending machine.

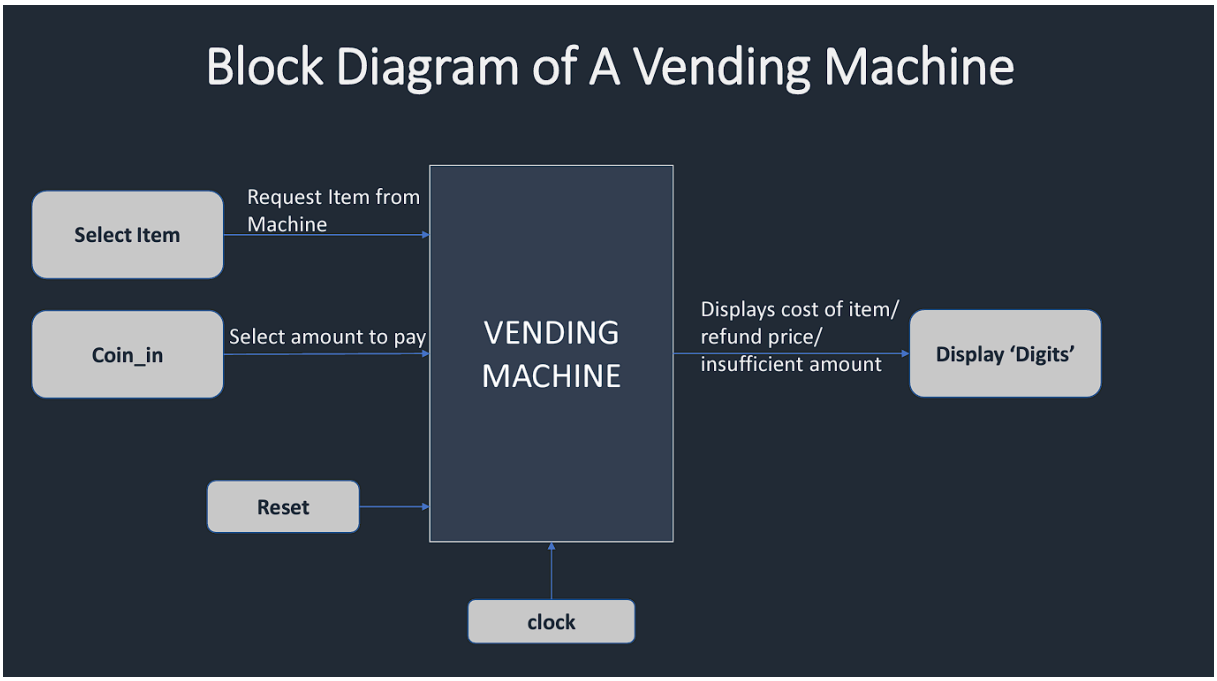


Figure 2: Vending Machine Block Diagram

3 Project Team management

We used Scrum which is a subset of agile methodology. Every week we would discuss the project's progress and come up with to-do tasks. In the beginning, the team had a planned meetings to decide what to work on during the upcoming week and the idea was discussed. We followed a collective way of solving different tasks in which each of us would work together at every step of the project.

The following were the tasks worked upon:

Brainstorming and concept creation: All team members worked together to brainstorm and formulate the initial idea for the smart greenhouse. Each team member contributed their unique perspectives and ideas to create a viable project plan.

Block Diagram and Circuit Design: All three members contributed to the design and development of the block diagram. Also, the schematic, PCB design and 3D view layout of the circuit on KiCAD was

accomplished as a collective effort. Each member brought their own ideas, and through discussion and iteration, they came up with the final diagrams.

Coding and Implementation: The coding and implementation of the vending machine concept and circuit design was a shared task. Each team member wrote sections of the code using different ideas to troubleshoot our issues. We worked together to ensure a compilation and simulation of the code in model SIM and proper synthesis and implementation of our code on the FPGA board.

4 Implementation

KiCAD: This tool is used to create the schematic for the vending machine circuit based on the functionalities as defined in the concept section. Here an evaluation board is given that contains libraries of the footprint and schematic for designing hardware with an FPGA board. This serves as the basis for designing the power supply, utilities and vending machine hardware circuit schematics as shown in figures 3, 4 and 5 respectively. In Figure 5, we make use of the switches and 7 segment display of the FPGA thus the reason for adding the symbols to the schematic. In order to generate the PCB layout, we must assign components to the symbols, by creating the footprint of the schematic, assigning components from the footprint libraries provided, Annotating components and performing ERC check.

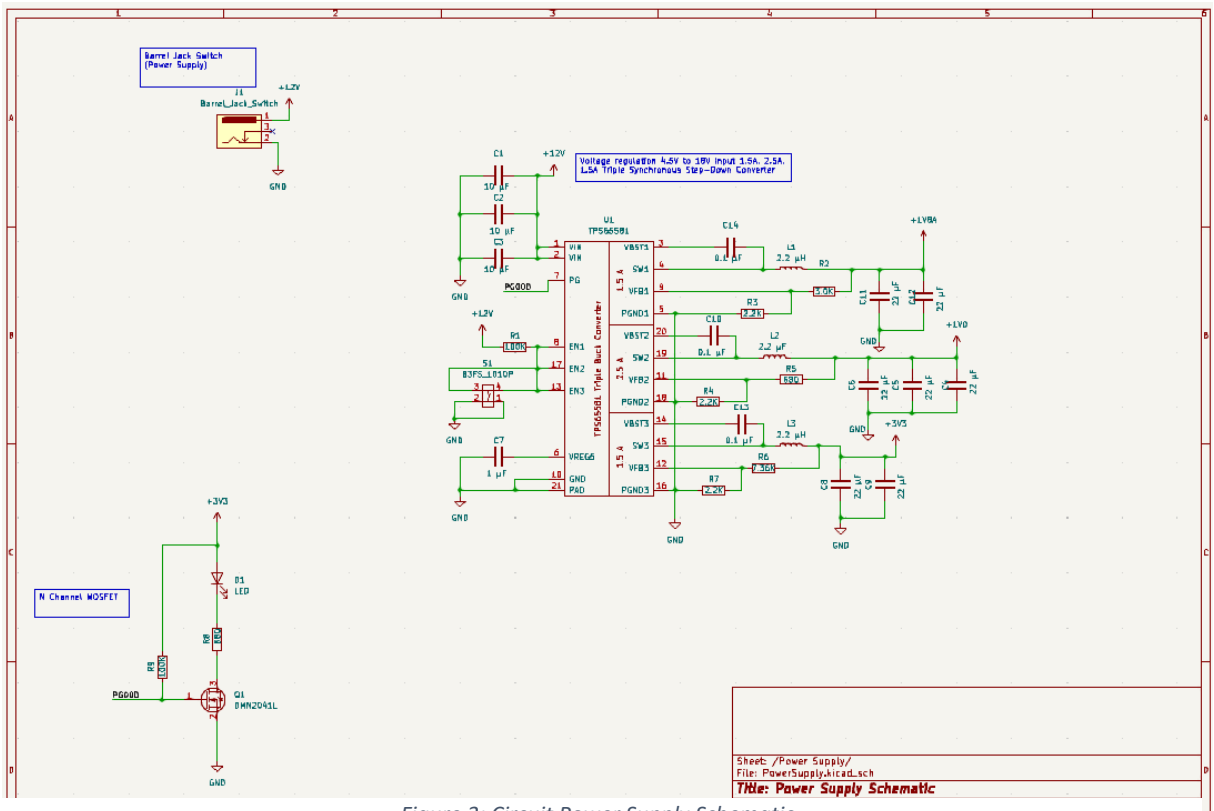


Figure 3: Circuit Power Supply Schematic

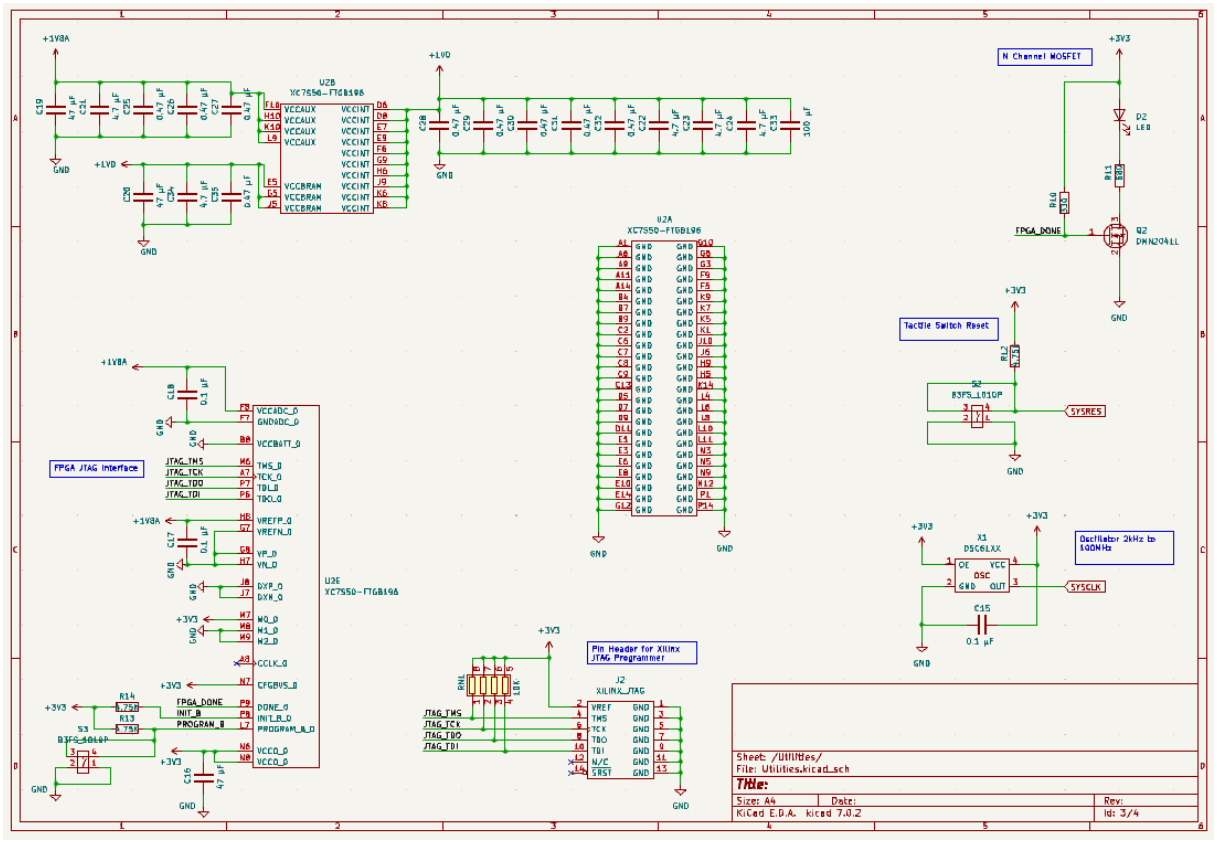
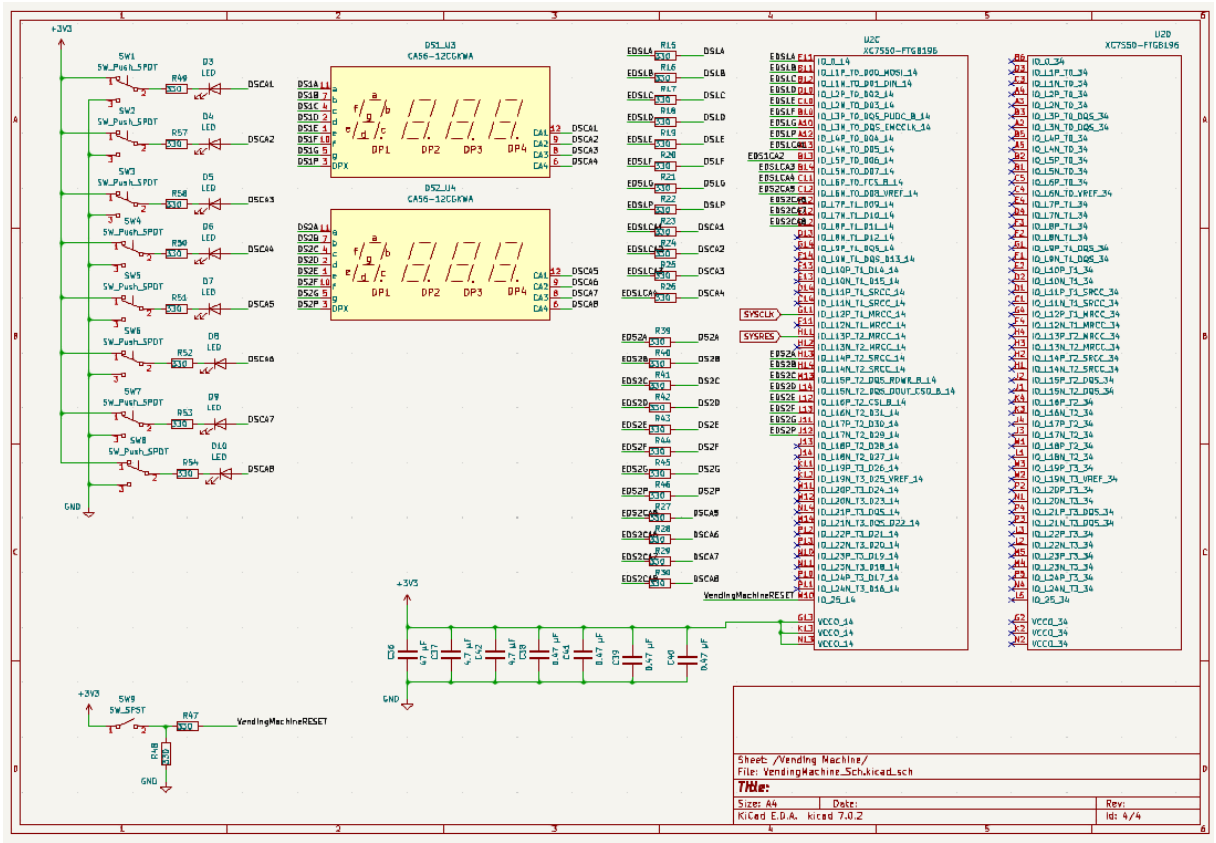


Figure 4: Circuit Utilities Schematic



These changes made in the schematic by assigning components are updated to the PCB, an outline (edge cut) is placed around as seen in figure 6, which corresponds to the size of the board followed by a DRC test. Auto-routing is performed to route the tracks on the board using an online free routing tool. A 3D view of the board can also be seen and saved as a .png file as shown in figure 7.

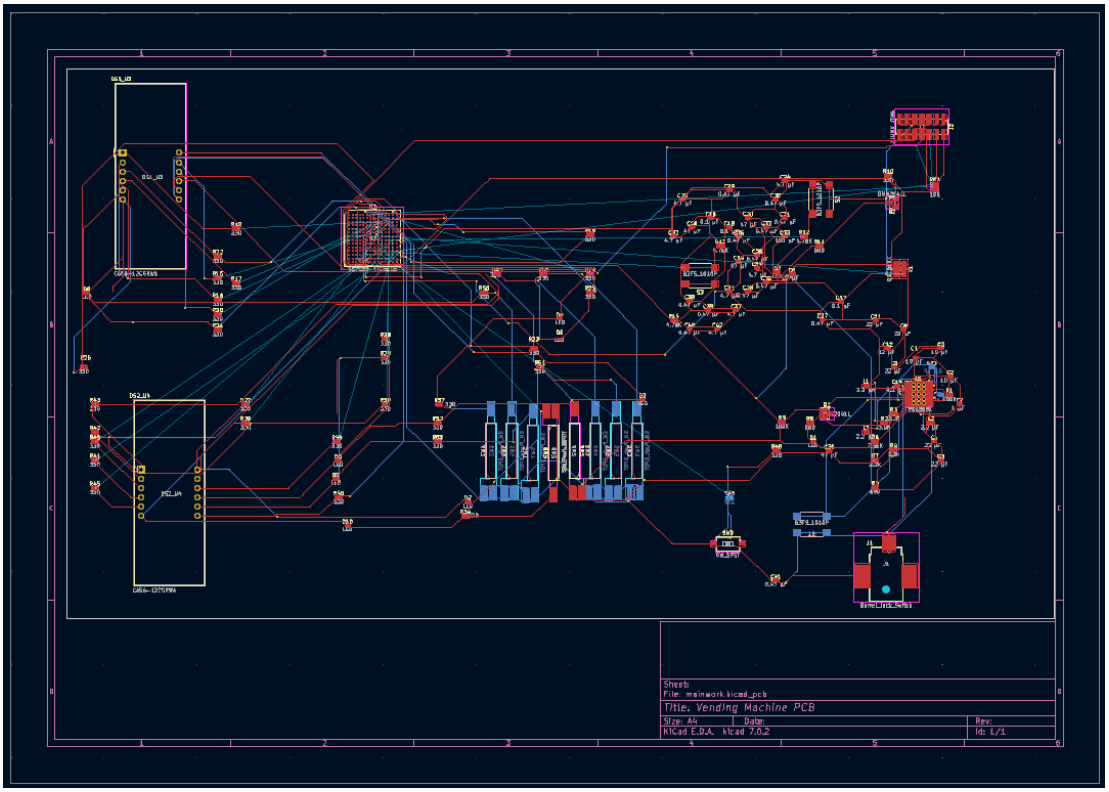


Figure 6: Figure 6: PCB Vending Machine with Auto Routing

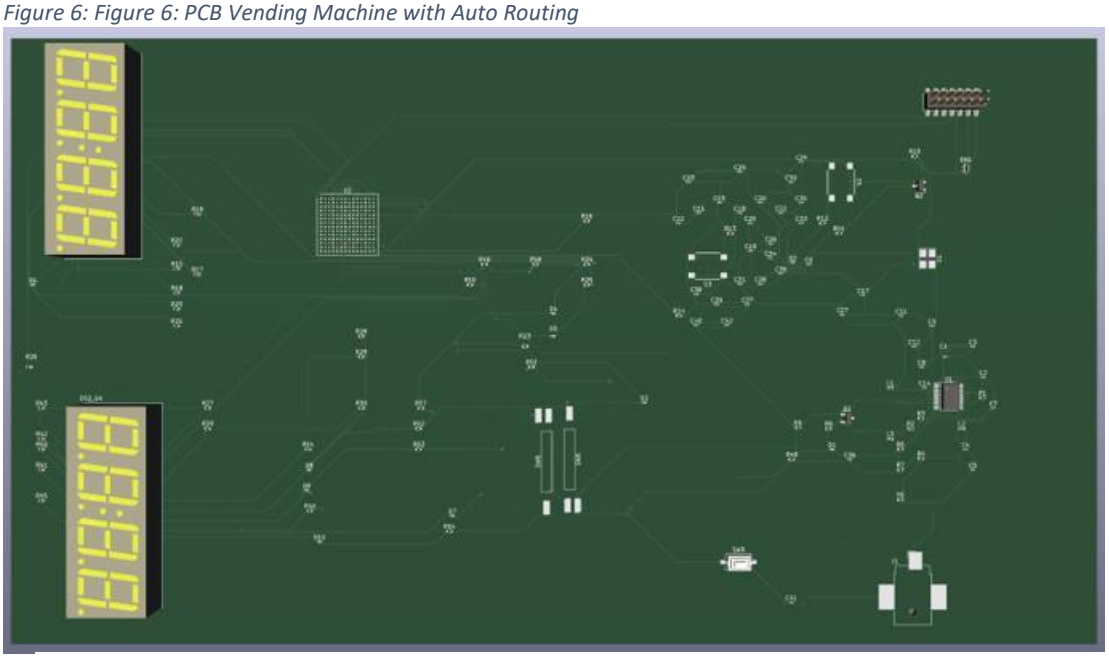


Figure 7: 3D view Vending Machine PCB

MODELSIM: Here the code for implementing the vending state machine is written. Three items are assumed to be contained in the vending machine. The users are expected to select an item and select the amount being placed into the machine. These items are being stored as products to identify the items being selected, and specific prices are assigned to each item. Signals are being used to determine the state of the machine from the idle state till an item has been dispensed from the machine. The DecoderBIt component enables the output values to be displayed on the 7-segment display as illustrated in Figures8, 9, and 10 respectively.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use ieee.numeric_std.all;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity VM is
7
8  port (
9   clk, reset: in std_logic;
10  coin_in, pickItem: in std_logic_vector(3 downto 0);
11 Anode, Cathode: out std_logic_vector(7 downto 0);
12 selectedItem, dispense: out std_logic_vector(3 downto 0);
13 refundOutLed, idle_state, selected_state, review_state, output_state: out std_logic
14 );
15
16 end entity;
17
18 architecture VMarch of VM is
19
20 signal total, refund, price, product : integer := 0;
21 type state is( idle, selected, review, output);
22 signal current_state, next_state : state;
23 signal clk_counter: std_logic_vector(19 downto 0);
24 signal counter_reset: std_logic_vector(3 downto 1);
25 signal Display: std_logic_vector(3 downto 0);
26
27
28 begin
29
30 memory: process (clk, reset)
31 begin
32
33     if (reset ='1') then
34         current_state <= idle;
35     elsif (clk 'event and clk = '1') then
36         current_state <= next_state;
37     end if;
38
39 end process;
```

```
41  logic: process (product,coin_in, pickItem, current_state, total, refund, price)
42  begin
43
44
45      case (current_state) is
46
47          when idle => if ( pickItem = "0001" ) then
48                                  next_state <=  selected;
49                       product <= 1;
50                       price <=4;
51                      selectedItem <="0001";
52                    elsif ( pickItem = "0010" ) then
53                                  next_state <=  selected;
54                       product <= 2;
55                       price <=3;
56                       selectedItem <= "0010";
57                    elsif ( pickItem = "0011" ) then
58                                  next_state <=  selected;
59                       product <= 3;
60                       price <=2;
61                       selectedItem <= "0011";
62                    else
63                      next_state <=  idle;
64                       product <= 0;
65                       price <=0;
66                       selectedItem <= "0000";
67                   end if;
68                  idle_state <= '1';
69                  selected_state <= '0';
70                  review_state <= '0';
71                  output_state <= '0';
72
73           when selected => if ( coin_in = "0101" ) then
74                       total <= 5;
75                        next_state <=  review;
76
77
78                    elsif ( coin_in = "0100" ) then
79                       total <= 4;
80                       next_state <=  review;
81
82                    elsif ( coin_in = "0011" ) then
83                       total <= 3;
84                       next_state <=  review;
85
86                    elsif ( coin_in = "0010") then
87                       total <= 2;
88                       next_state <=  review;
89                        else
90                       total <= 0;
91                       next_state <=  selected;
92                    end if;
93                  idle_state <= '1';
94                  selected_state <= '1';
95                  review_state <= '0';
96                  output_state <= '0';
97
```

```
98          when review => if  (total >= price) then
99                      refund <= total - price;
100                     refundOutLed <= '1';
101                     next_state <=  output;
102                 else
103                     refund <= 0;
104                     refundOutLed <= '0';
105                     next_state <=  selected;
106
107                   end if;
108                 idle_state <= '1';
109                 selected_state <= '1';
110                 review_state <= '1';
111                 output_state <= '0';
112
113         when output =>
114                     if (product = 1) then
115                       dispense <= "0001";
116                           next_state <=  idle;
117                     elsif  (product = 2) then
118                       dispense <= "0010";
119                           next_state <=  idle;
120                     elsif  (product = 3) then
121                       dispense <= "0011";
122                           next_state <=  idle;
123                     else
124                       next_state <=  selected;
125                     end if;
126                 idle_state <= '1';
127                 selected_state <= '1';
128                 review_state <= '1';
129                 output_state <= '1';
130         when others =>
131                   next_state <=  idle;
132                     product <= 0;
133                     price <=0;
134                     refund <= 0;
135                     refundOutLed <= '0';
136
137                     idle_state <= '0';
138                   selected_state <= '0';
139                   review_state <= '0';
140                   output_state <= '0';
141         end case;
142 end process;
143
```

When a user inputs the value for money being used in purchasing an item. When the value amount is more than the price of the item, the machine displays the refund value, else it waits for the user to put in the remainder amount. The display then shows the item being dispensed.

VIVADO: This is an environment used for synthesizing, implementing, routing design and, generating bit stream of the VHDL code that would be uploaded to the FPGA board to perform the designed logic. Below the code for displaying the vending machine signals and states are shown.

```
144     ------------------------------------------------
145  --Display
146   reset_counter: process(clk, reset)
147  begin
148      if (reset = '1') then
149          clk_counter <= (others => '0');
150      elsif (clk 'event and clk = '1') then
151          clk_counter <= clk_counter + 1;
152      end if;
153  end process;
154
155  -- Activating the anode at the reset counts of the clk_counter
156  --and displaying the values to be decoded by the cathode signal
157  counter_reset <= clk_counter(19 downto 17);
158  Activate_Anode: process(counter_reset, refund)
159  begin
160      case (counter_reset) is
161          when "000" =>
162              Anode <= "01111111";
163          if (pickItem = "0001" ) then
164              Display <= "0100";
165          elsif (pickItem = "0010" ) then
166              Display <= "0011";
167          elsif (pickItem = "0100" ) then
168              Display <= "0010";
169          else
170              Display <= "0000";
171          end if;
172
173          when "001" =>
174              Anode <= "11011111";
175          if (coin_in = "0101" ) then
176              Display <= "0101";
177          elsif (coin_in = "0100"  ) then
178             Display <= "0100";
179          elsif (coin_in = "0100" ) then
180              Display <= "0011";
181          elsif (coin_in = "0010" ) then
182             Display <= "0010";
183          else
184              Display <= "0000";
185          end if;
186
```

```
187         when"010" =>
188             Anode <= "11111011";
189         if (total >= price ) then
190             Display <= std_logic_vector(to_unsigned(refund, display'length));
191         else
192             Display <= "0000";
193         end if;
194 
195         when "100" =>
196                 Anode <= "11110111";
197         if (product = 1) then
198             Display <= "0001";
199         elsif (product = 2) then
200             Display <= "0010";
201         elsif (product = 3) then
202             Display <= "0100";
203         else
204             Display <= "0000";
205         end if;
206 
207         when "101" =>
208             Anode <= "11111101";
209           if (pickItem /= "0000")then
210             Display <= "1011";
211         elsif (coin_in /= "0000") then
212             Display <= "1010";
213         elsif (refund >= 0) then
214             Display <= "1000";
215         else
216             Display <= "0110";
217         end if;
218         when others =>
219             Anode <= "11111110";
220             Display <= "0000";
221     end case;
222 end process;
223 
```

A test bench is created in order to verify and validate that our logic wors correctly [see in GitHub].
Next, we allocate the I/O pins from the constraint file to the signal parameter of the vending machine logic [see figure 8].

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
#create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {CLK100MHZ}];


##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { pickItem[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { pickItem[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { pickItem[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { pickItem[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
#set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { coin_in[0] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { coin_in[1] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { coin_in[2] }]; #IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { coin_in[3] }]; #IO_25_34 Sch=sw[9]

## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { selectedItem[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { selectedItem[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { selectedItem[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { selectedItem[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
#set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { dispense[0] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { dispense[1] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { dispense[2] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { dispense[3] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { idle_state }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { selected_state }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { review_state }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { output_state }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { refundOutLed }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]

#7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { Cathode[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { Cathode[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { Cathode[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { Cathode[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { Cathode[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { Cathode[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { Cathode[6] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { Cathode[7] }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { Anode[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { Anode[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { Anode[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { Anode[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { Anode[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { Anode[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { Anode[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { Anode[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]

##Buttons
#set_property -dict { PACKAGE_PIN C12   IOSTANDARD LVCMOS33 } [get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn
#set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTNC }]; #IO_L9P_T1_DQS_14 Sch=btnc
set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { reset }]; #IO_L4N_T0_D05_14 Sch=btnu
```

Figure 8: Activated Ports in constraint file VIVADO

Finally we upload the code to the FPGA by converting the vending machine logic to gates (running synthesis), Implementation of the logic where the route design is shown as seen in Figure 9, generating the bitstreams, and finally programming the FPGA.

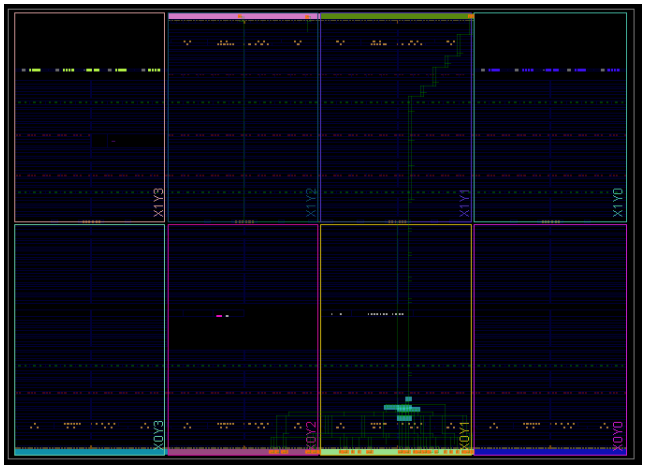


Figure 9: Route Design Vending Machine

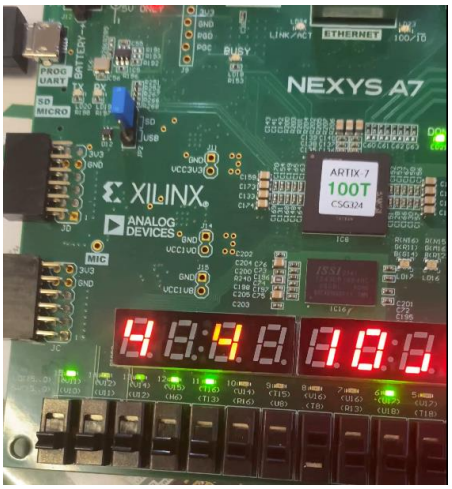


Figure 10: FPGA DISPLAY