LAB E5-VHDL DESIGN PROJECT

EEE20001 Digital Electronics Design

1.	**Student Name:**	**Sze XiJie**	**Student ID:**	**101222928**
2.	**Student Name:**	**Sarah Lau Li Tin**	**Student ID:**	**101220896**
3.	**Student Name:**	**Lee Kai Ze**	**Student ID:**	**101222041**

Lab Supervisor Name: **Dr. Tay Fei Siang**

MAY 12, 2021

Swinburne University of Technology Sarawak

Table of Contents

Brief Description 3
Description of Approach 3
Block Diagram of Top-Level Structure 4
State Transition Diagram (STD) 9
Detailed Description of Modules 12
SyncButton Module 12
TrafficLights Module 12
Timer Module 13
MotorController Module 13
TimerReset Module 13
Detailed Description of Simulation 14
SyncButton Module 14
TrafficLights Module 20
Timer Module 24
MotorController Module 26
TimerReset Module 30
Detailed Description on Debugging Process 34
Stepper Motor Will Not Stop Rotating After 5 rotations 34
Traffic Light Flashing Is Not Consistent and Accurate 35
Listing of VHDL Modules 36
ISE Synthesis Report 52
Appendix 60

Figure 1: Block Diagram of Top-Level Structure 4
Figure 2: Block Diagram of TImer Module 4
Figure 3: Block Diagram of TrafficLights Module 5
Figure 4: Block Diagram of SyncButton Module 6
Figure 5: Block Diagram of MotorController Module 7
Figure 6: Block Diagram of TimerReset Module 8
Figure 7: Flowchart of Project 9
Figure 8: State Diagram for MotorController Module 10
Figure 9: State Diagram for TrafficLights Module 10
Figure 10: State Diagram for TimerReset Module 11
Figure 11: State Diagram for SyncButton Module 11
Figure 12: TB_SyncButton TestBench Code 18
Figure 13: SyncButton Module Simulation at 0ns to 1500ns 19
Figure 14: SyncButton Module Simulation at 1500ns to 3000ns 19
Figure 15: TB_TrafficLights TestBench Code 22

Figure 16: TrafficLigths Module Simulation 23
Figure 17: TrafficLights Module Simulation at 200ns to 300ns 23
Figure 18: TB_Timer TestBench Code 25
Figure 19: Timer Module Simulation 25
Figure 20: TB_MotorController TestBench Code 28
Figure 21: MotorController Module Simulation 29
Figure 22: MotorController Module Simulation at 700ns to 800ns 29
Figure 23: MotorController Module Simulation at 900ns to 1000ns 29
Figure 24: TB_ TimerReset TestBench Code 32
Figure 25: TimerReset Module Simulation 32
Figure 26: MotorController Module Code 39
Figure 27: Timer Module Code 41
Figure 28: TrafficLights Module Code 44
Figure 29: TimerReset Module Code 47
Figure 30: SyncButton Module Code 51
Figure 31: ISE Sysnthesis Report 58
Figure 32: ISE Summary Report 59
Figure 33: TrafficTopLevel.vhd Code 62
Figure 34: TopLevel.ucf Code 63

Brief Description

Description of Approach

The aim of the project is to design a traffic light controller which is responsible for controlling the traffic light to have the train as the top priority. It is also required to simulate the gate closing and opening by enabling and rotating the Stepper Motor for 5 rotations to completely close and open and gate. To complete this project, 5 modules are created: SyncButton Module, TrafficLights Module, Timer Module, TimerReset Module and MotorController Module.

SyncButton Module is created to sync all the button inputs. After synchronizing all the inputs, the priorities will also be set to have the train as the top priority, regardless of other button inputs pressed when the train is passing. If the other button inputs are pressed, and the train button is pressed afterwards, the train will pass through while the car and pedestrian will need to wait until the train has passed.

Next, the TrafficLights Module is created to control the traffic lights and also provide an input for the MotorController Module. When train button is pressed, traffic lights will flash red and amber, if car button is pressed, the traffic lights will be green and when the pedestrian button is pressed, the pedestrian light will be on and the traffic lights will be green.

The MotorController Module is created to control the Stepper Motor. By getting the outputs form the TrafficLights Module and the SyncButton Module, the Stepper Motor will enable and rotate in clockwise or counter-clockwise direction depending on the states. If gate is closing, the Stepper Motor will rotate clockwise, while counter-clockwise when the gate is opening.

The Timer Module is created to simulate a timing system for the project. The Timer Module will generate 4Hz pulses for the TrafficLights Module to let the traffic lights flash red and amber, generate motor pulse for the MotorController Module to rotate the Stepper Motor and to generate a counter to keep track of when is the right time to enable or disable the Stepper Motor and to hold the train and pedestrian passing time.

Lastly is the TimerReset Module which is created to reset the Timer Module. This is required to reset the counter for the gate opening and closing to only let it rotate for 5 rotations.

Block Diagram of Top-Level Structure

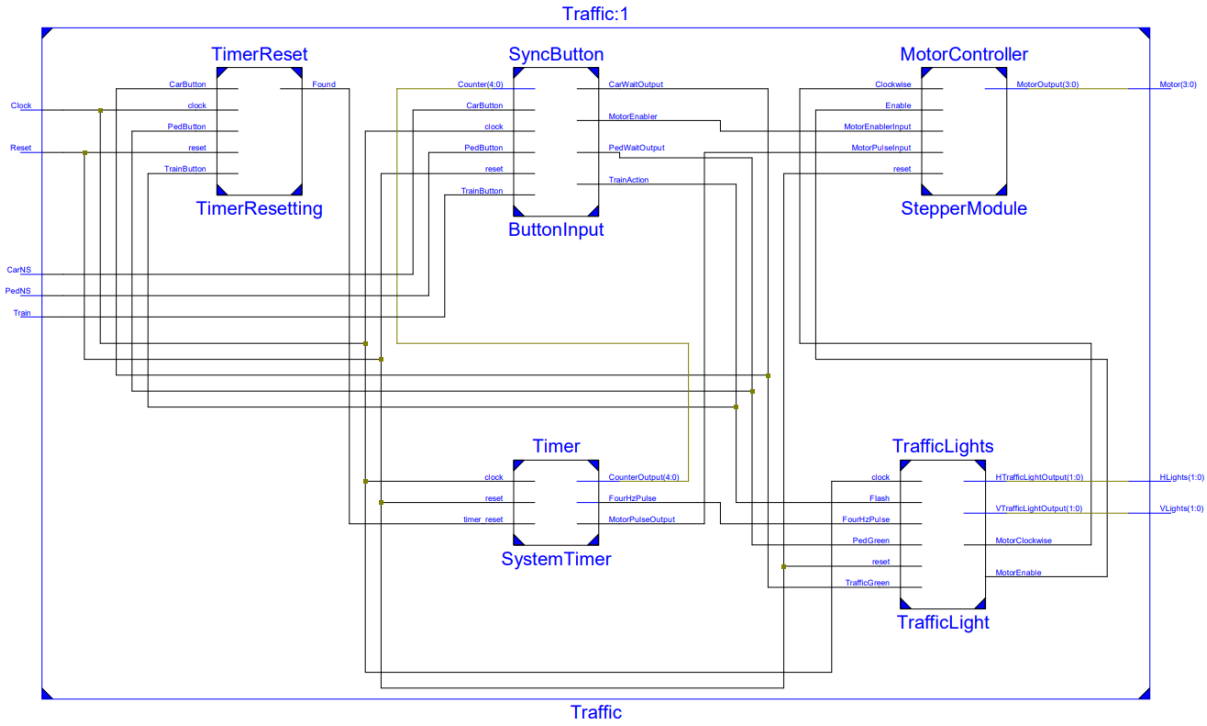

Figure 1: Block Diagram of Top-Level Structure

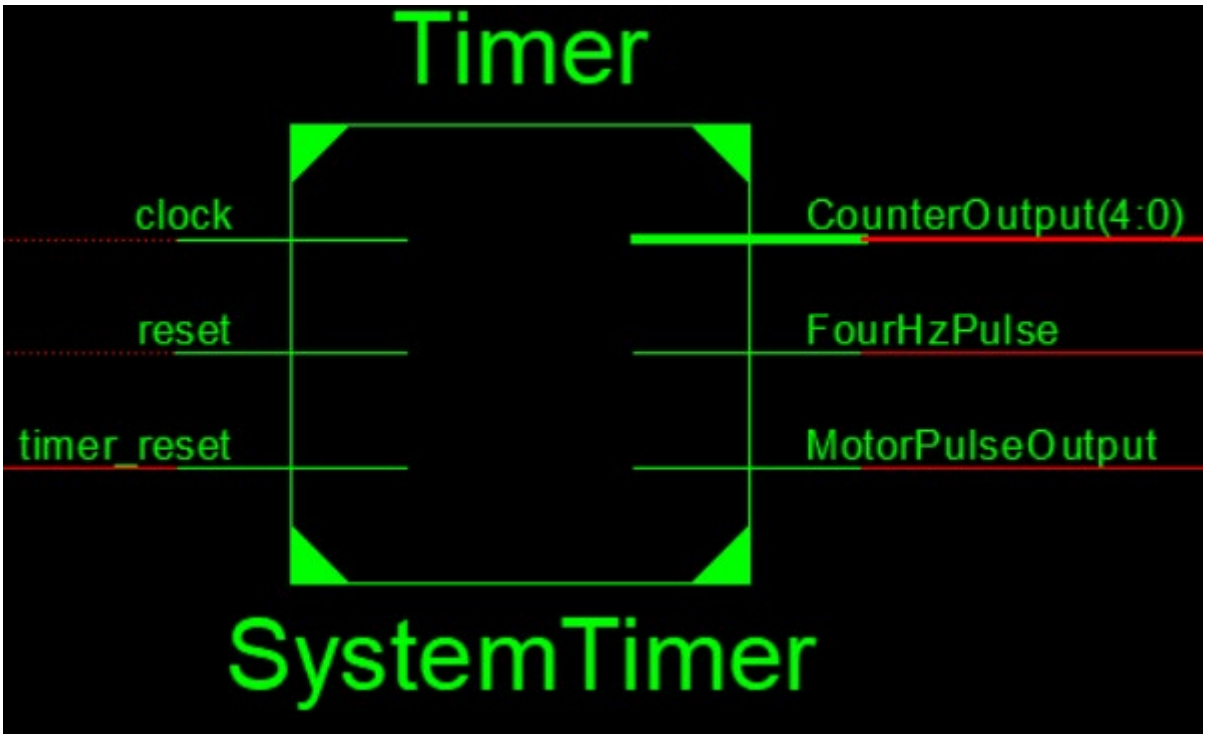

Figure 2: Block Diagram of TImer Module

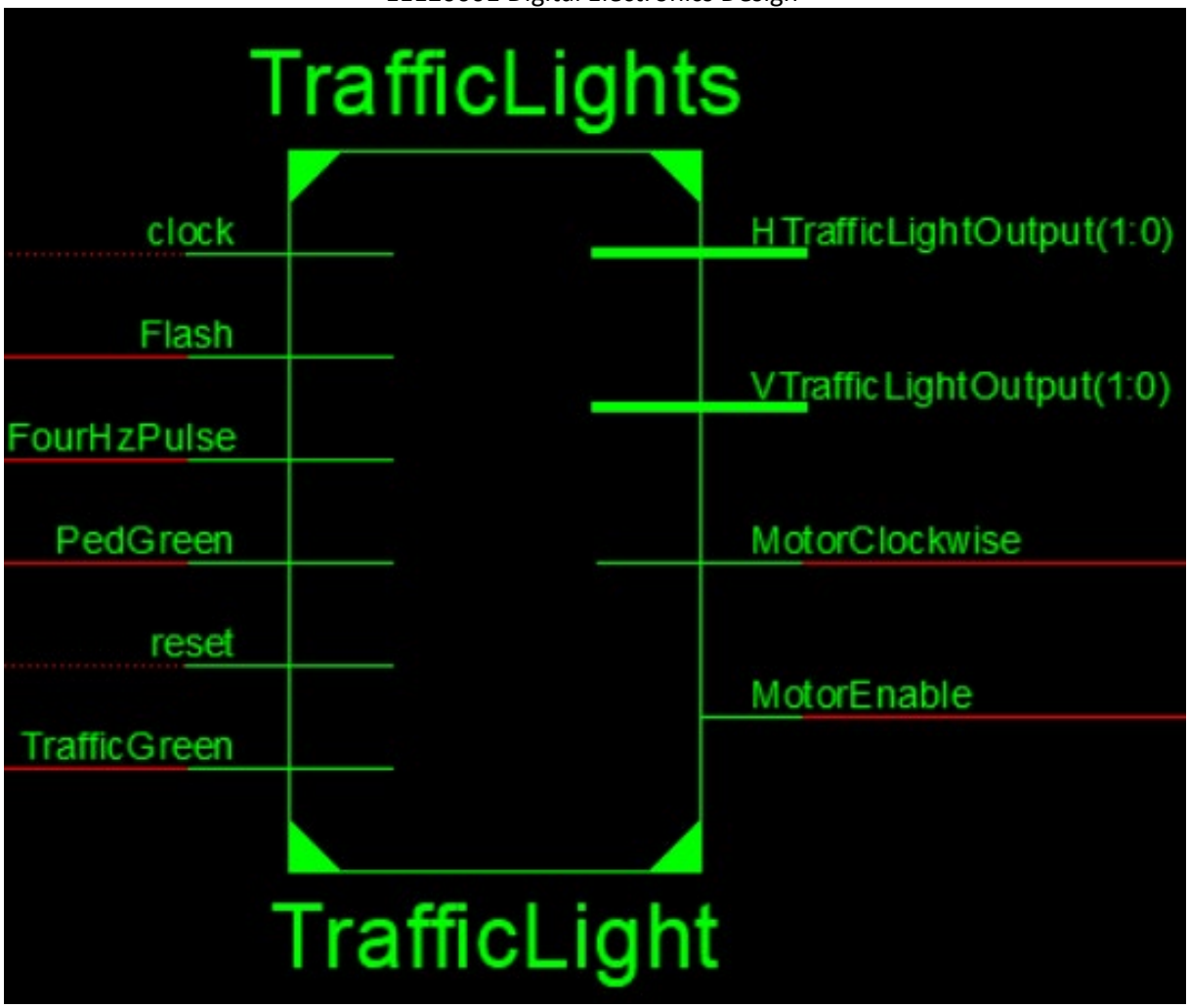

Figure 3: Block Diagram of TrafficLights Module

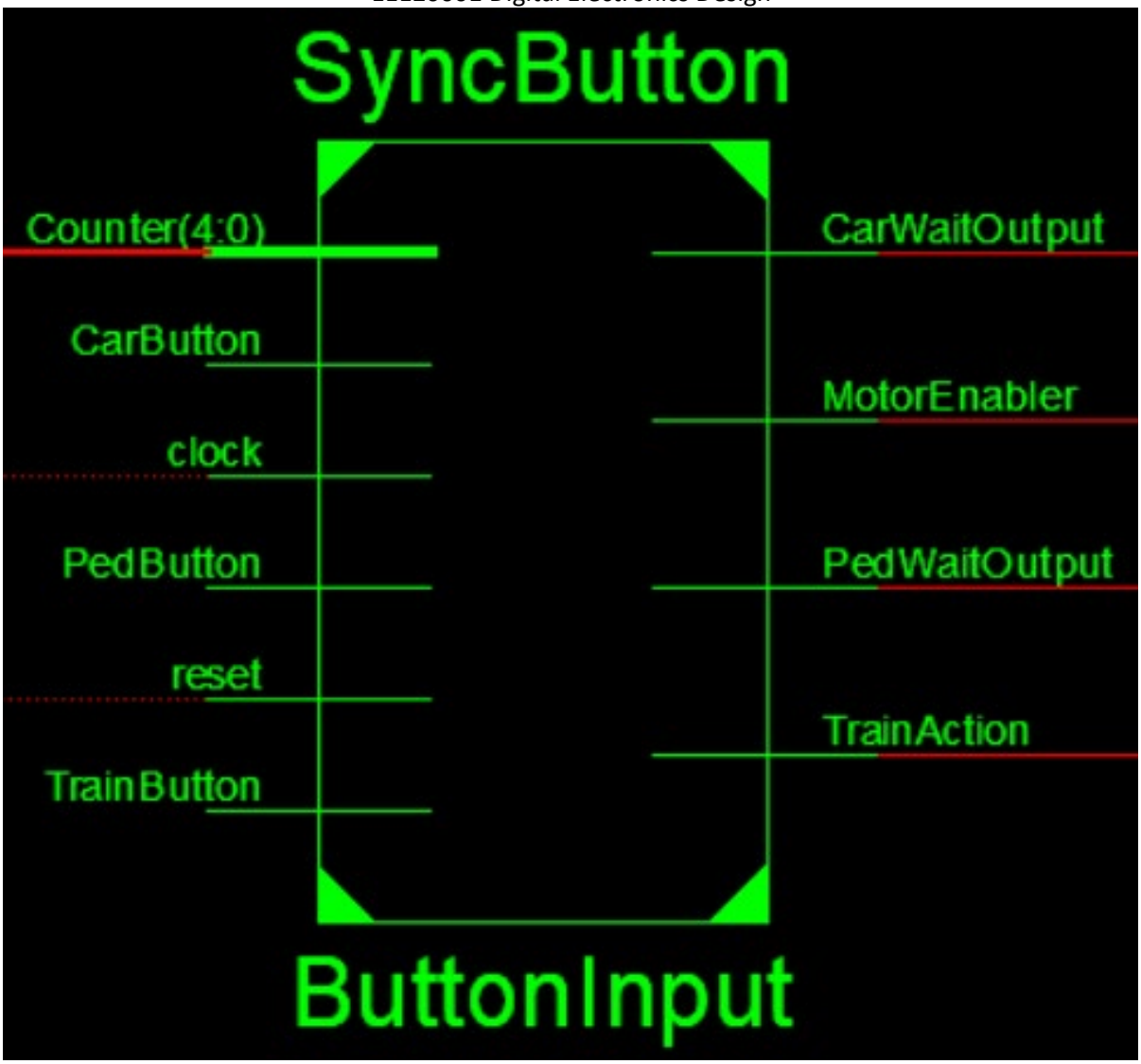

Figure 4: Block Diagram of SyncButton Module

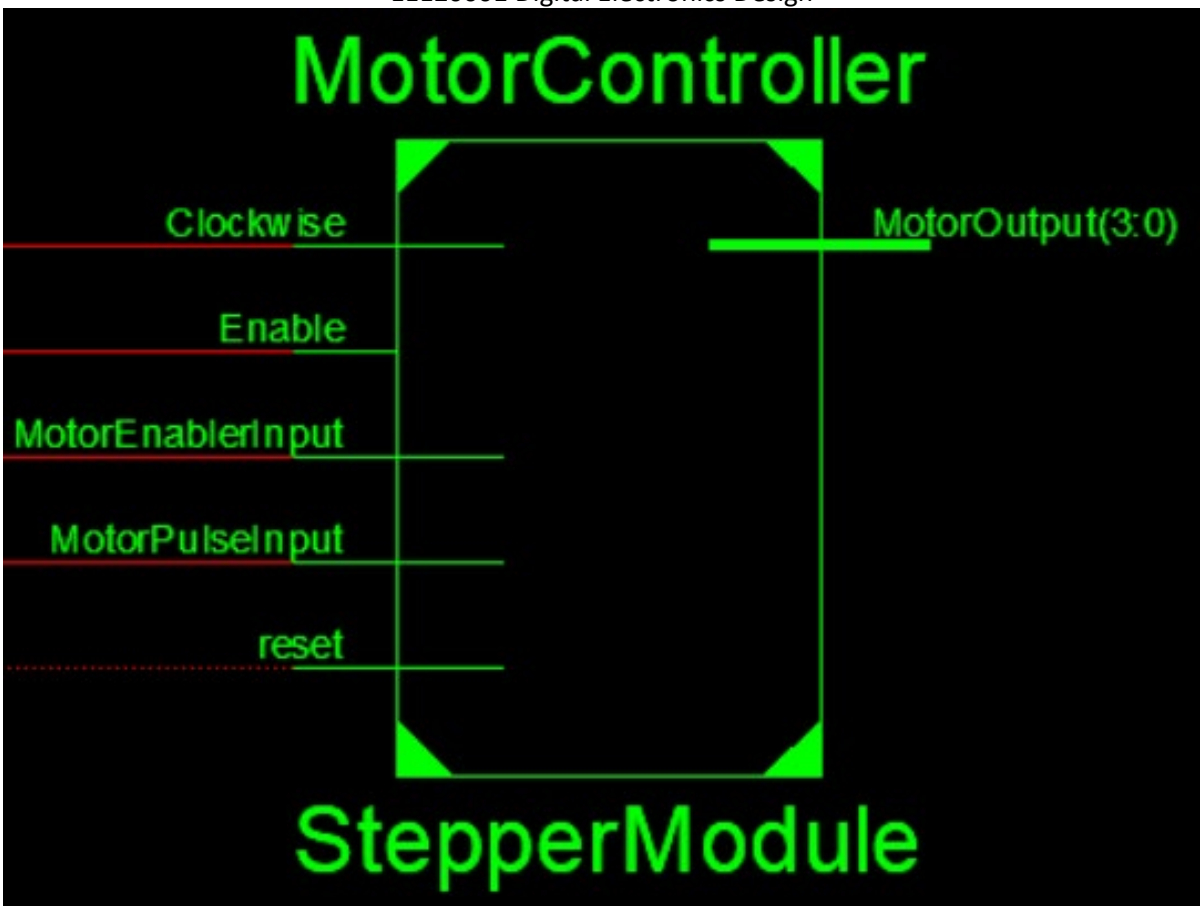

Figure 5: Block Diagram of MotorController Module

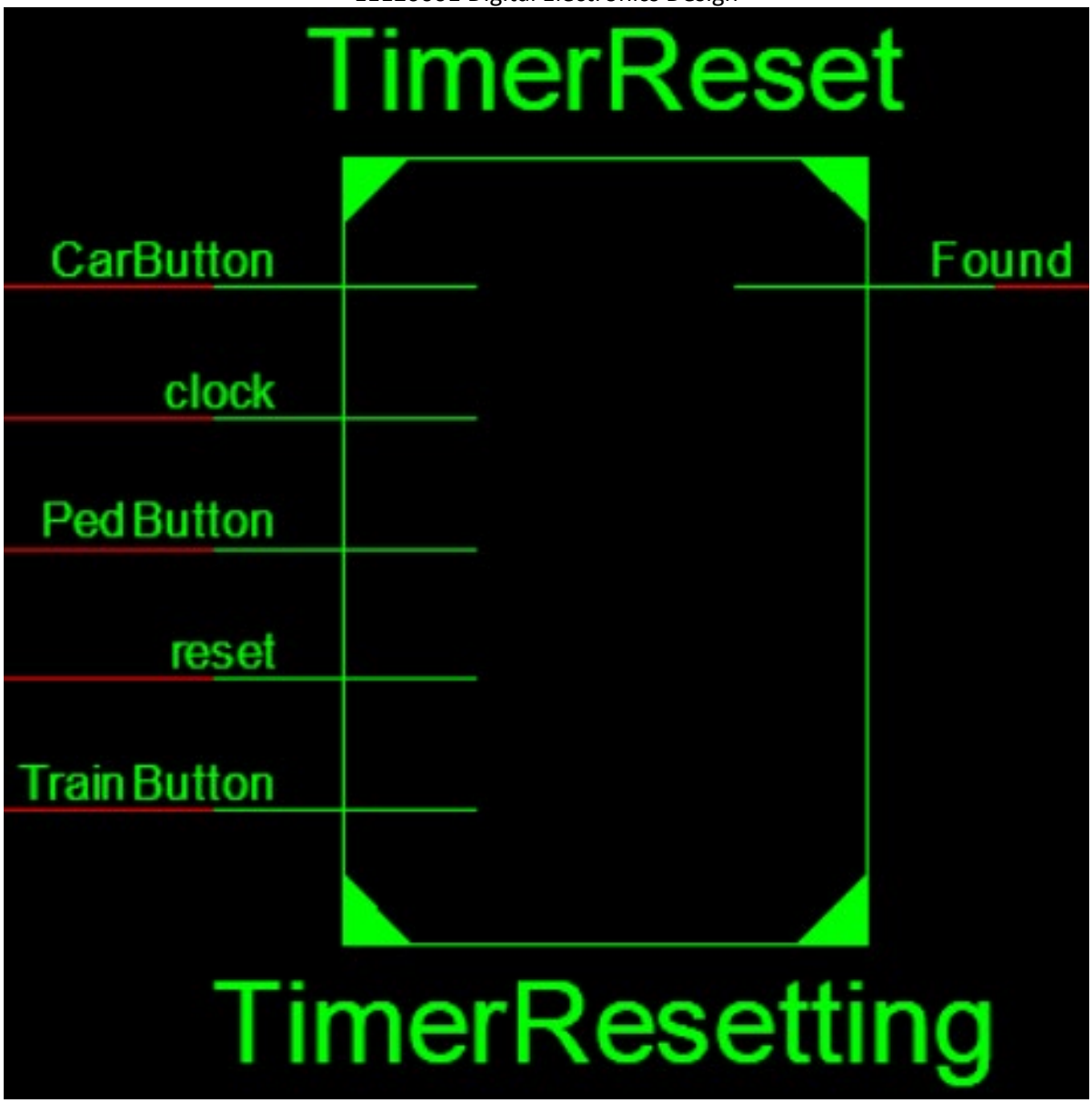

Figure 6: Block Diagram of TimerReset Module

State Transition Diagram (STD)

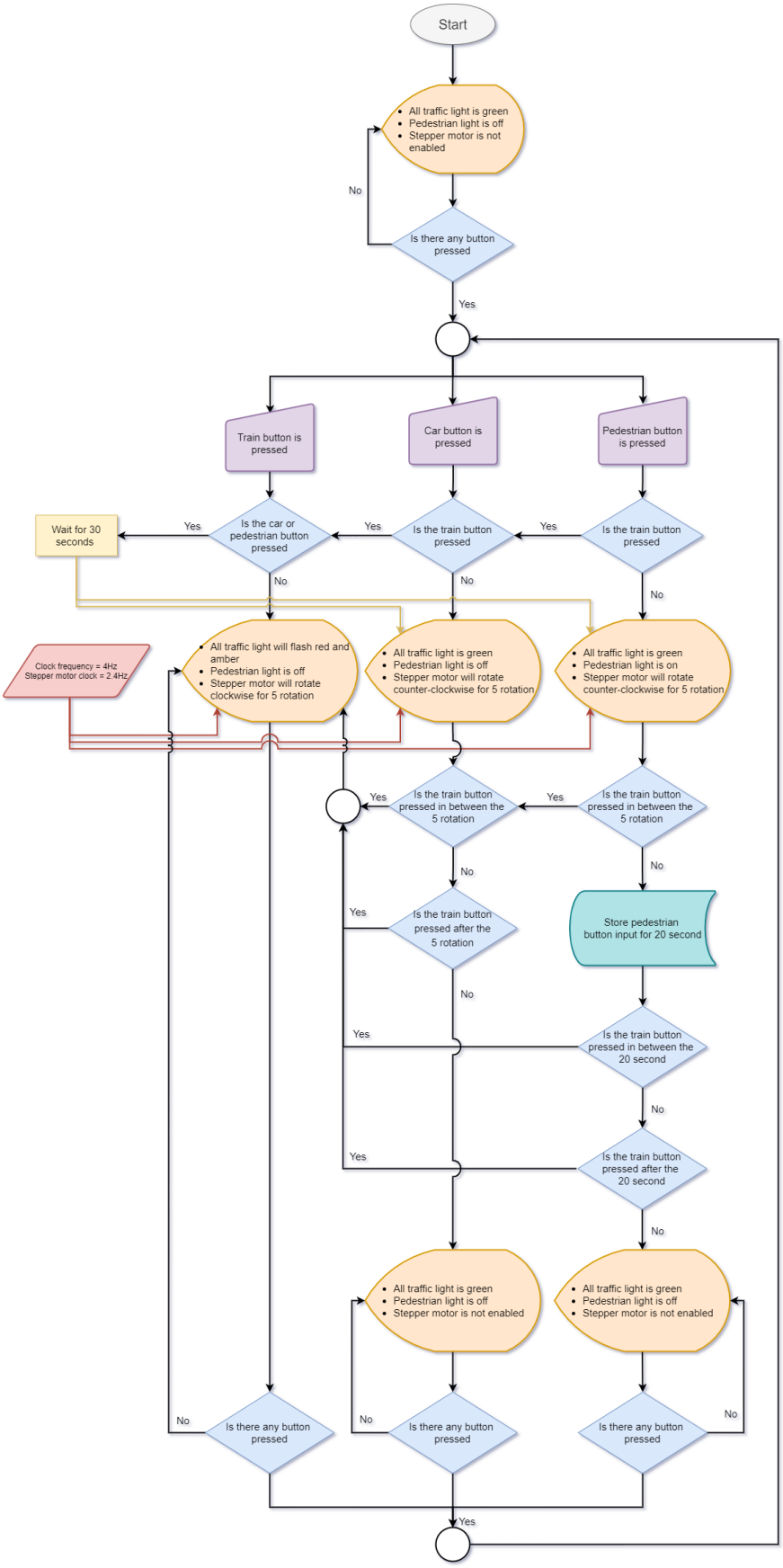

Figure 7: Flowchart of Project

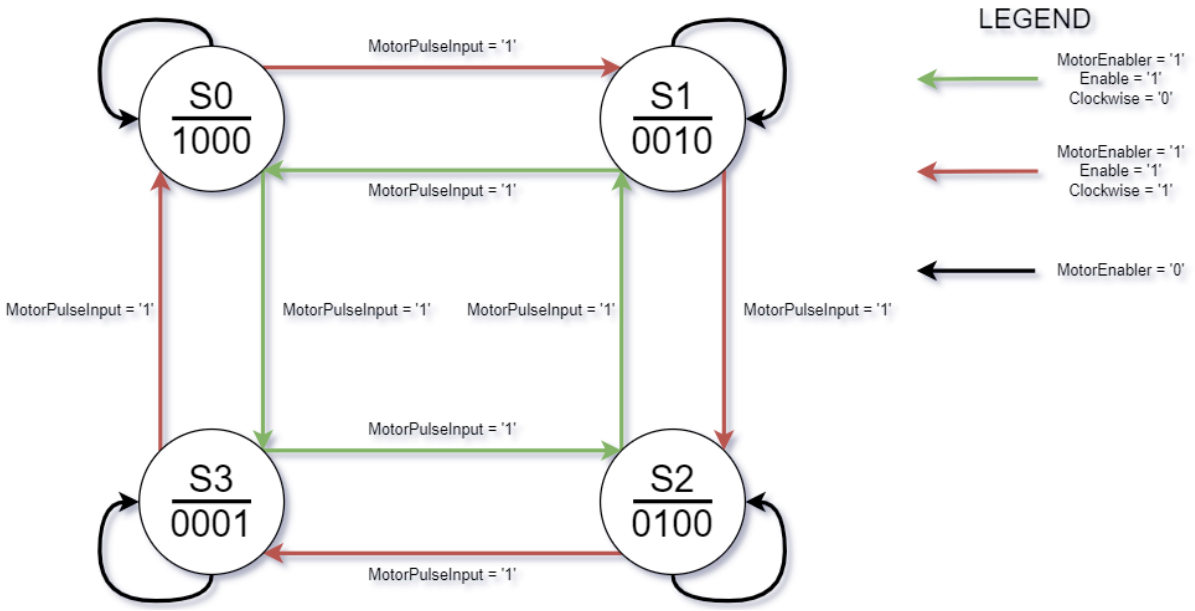

Figure 8: State Diagram for MotorController Module

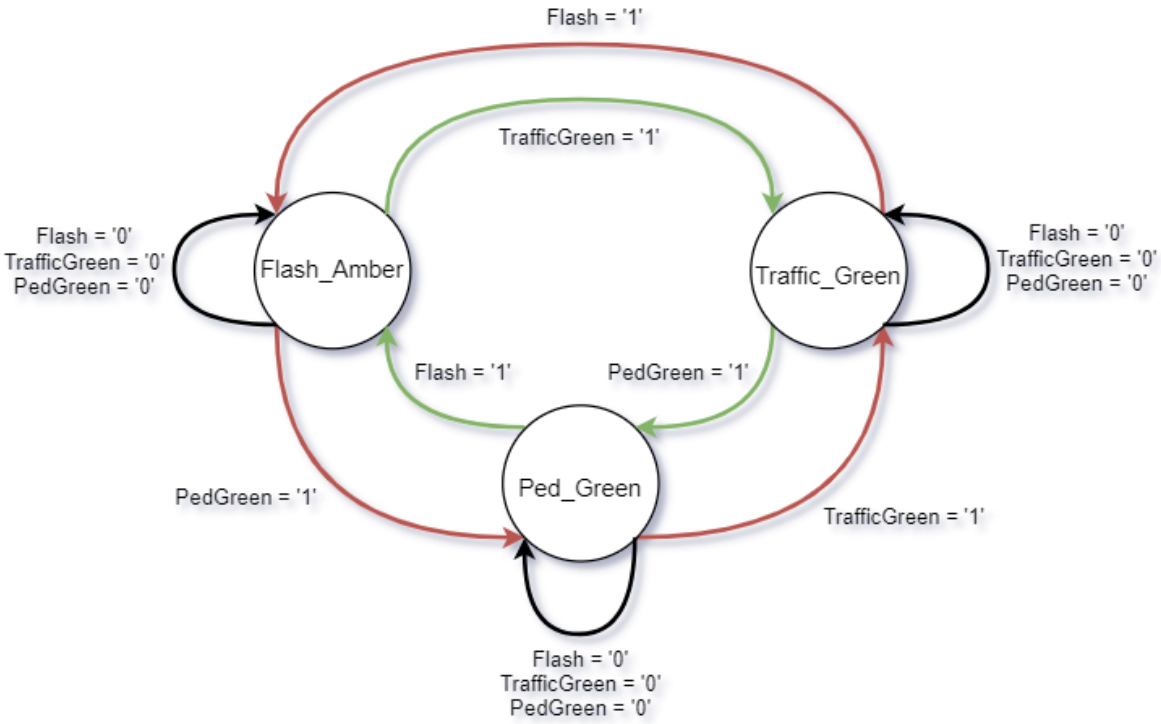

Figure 9: State Diagram for TrafficLights Module

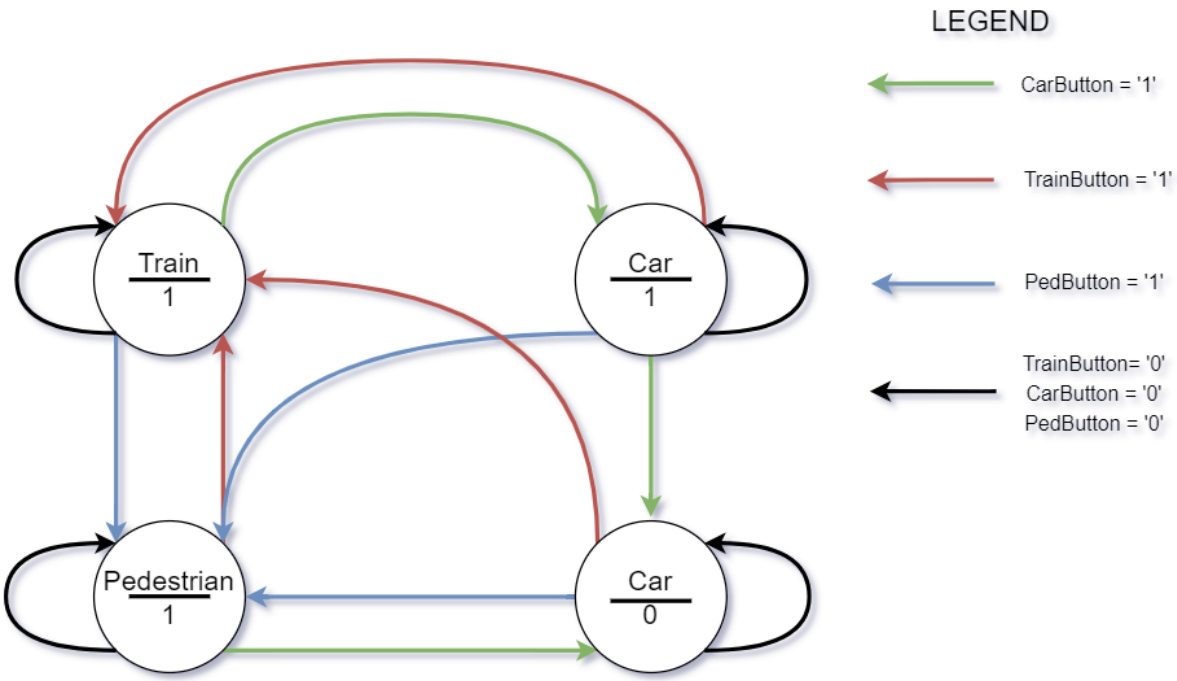

Figure 10: State Diagram for TimerReset Module

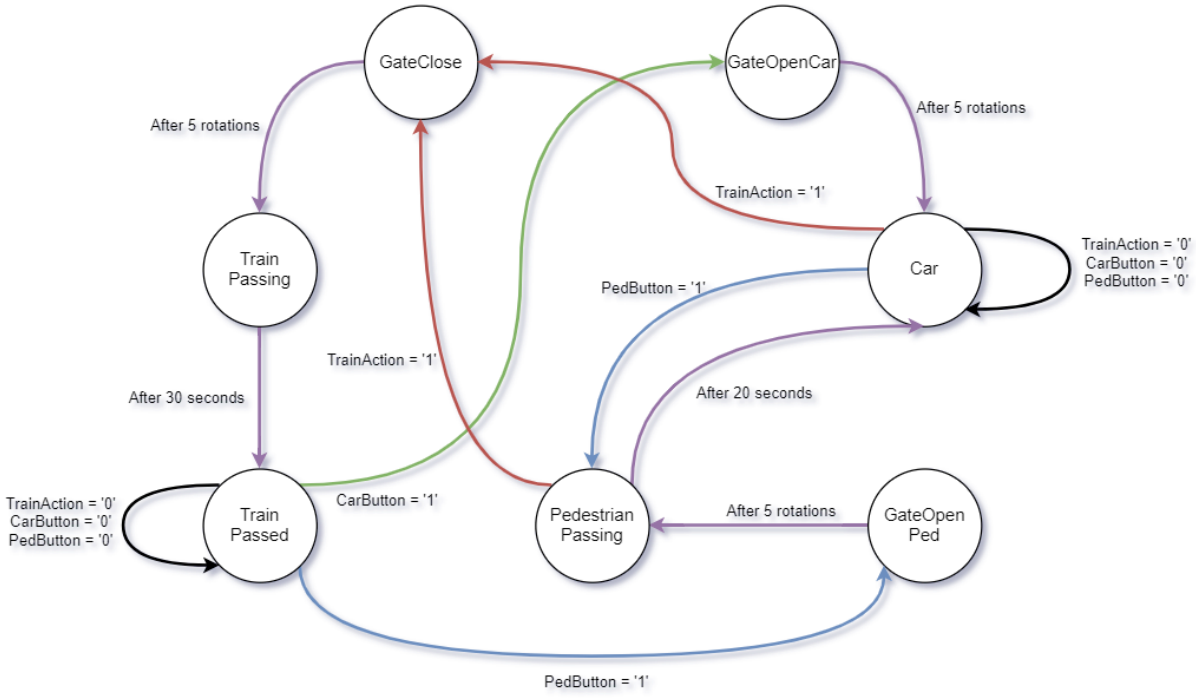

Figure 11: State Diagram for SyncButton Module

Detailed Description of Modules

For all the modules, reset is connected to the Master Reset which is a button from the Button Module and the clock will be from the Clock Module.

SyncButton Module

The SyncButton Module is used to sync the button pressed and to identify when to enable the Stepper Motor. In the SyncButton Module, a Moore Machine is used to give an accurate output. 7 states will be used in the Moore Machine, which is GateClose, GateOpenCar, GateOpenPed, TrainPassing, TrainPassed, Car, PedestrianPassing states. Initailly, the state will be Car when the master reset is pressed. Then when it detects the train button is pressed, it will move to GateClose state to close the gate. After it has closed the gate, it will move to TrainPassing state to simulate the train is passing. After it has finished, it will move to TrainPassed state to simulate that the train has passed. In GateClose and TrainPassing states, when the car or pedestrian button is pressed, it will be ignored as the train takes priority. It will only receive the button inputs when it is in TrainPassed state. So, during that state, when car button is pressed, it will move to GateOpenCar to open the gate, then after gate is fully opened, it will move to Car state. Now, when a pedestrian button is pressed, the gate will not open again as it is already opened, and it will directly move to PedestrianPassing state and hold the input for 20 seconds to let the pedestrian cross the road. If a train button is pressed in between the 20 seconds, the gate will start to close again. Similarly, instead of car button pressed when the state is at TrainPassed state, if a pedestrian button is pressed, the state will move to GateOpenPed and then move to PedestrianPassing state. During GateClose, GateOpenCar, GateOpenPed, MotorEnablerOutput will be '1' to enable the Stepper Motor and at TrainPassing, TrainPassed, Car, PedestrianPassing states, the MotorEnablerOutput will be '0', to disable the Stepper Motor as it has finished rotating 5 times. TrainAction, CarWaitOutput and PedWaitOutput will output accordingly when the corresponding button is pressed. Refer to Figure 30: SyncButton Module Code for full view of code.

TrafficLights Module

The TrafficLights Module is used to control the traffic lights and the pedestrian lights. The TrafficLights Module also uses Moore Machine to determine the output for the traffic lights and Stepper Motor. Initially, the state will be set at Traffic_Green state. When the train button is pressed, the state will move to Flash_Amber state. In this state, the traffic lights will flash red and amber according to the 4Hz pulse and the Stepper Motor will rotate in a clockwise direction to simulate the closing of the gate. If the car button is pressed, the state will move back to Traffic_Green state. The Stepper Motor will rotate counter-clockwise to simulate gate opening and the traffic lights will be green. Similarly, in Flash_Amber state, if the pedestrian button is pressed, the state will move to PedGreen state where the Stepper Motor will also rotate counter-clockwise to simulate opening of gate and the traffic lights will be green and pedestrian light will be on. The same goes to when the pedestrian button is pressed in Traffic_Green state. During PedGreen state, if the train button is pressed, the state will move to Flash_Amber state. The reason why there is no code to check if the pedestrian or train has finished its action based on the timer is because this action is done in SyncButton Module. Therefore, the TrafficLights Module only needs to output for the traffic light, pedestrian lights, and the motor rotation. Refer to Figure 28: TrafficLights Module Code for full view of code.

Timer Module

For Timer Module, there will be reset, timer_reset and rising clock edge. The reset will be for the master reset which is a button input from the hardware to reset everything, the timer_reset will be to reset the timer whenever a state change from train to car/pedestrian or car/pedestrian to train occurs. And if none of the above happens, timer will increment based on the rising clock edge. So, whenever the counter reaches "1111101000" which is 1 seconds, the timekeeper will increment by 1. Then, when the counter reaches the same binary as the MotorCount, then MotorPulseOutput will output 1 and MotorCount will increment by "0001010011" which is 83 pulses to determine the next MotorPulseOutput or else MotorPulseOutput will output 0. Refer to Figure 27: Timer Module Code for full view of code.

MotorController Module

For the MotorController Module, it is simple. The module consists of a moore machine which will change states based on the MotorEnablerInput, Enable and Clockwise inputs. Initially, the state will be reset to S0 which will output "1000" for the Stepper Motor and following the rising clock edge of the MotorPulseInput, it will change states. When MotorEnablerInput is '0', the Stepper Motor will not be enabled for all states, thus it will hold the current states until MotorEnablerInput is '1'. MotorEnablerInput essentially lets the Stepper Motor to rotate 5 rotations then disabling it to simulate the gate finished closing or opening. When MotorEnablerInput is still '1', if Enable is '0', the Stepper Motor still will not be enabled until is it '1', when it is '1', depending on the Clockwise input, it will either rotate clockwise or counter-clockwise. When Clockwise is '1', the Stepper Motor will rotate clockwise, and the state change will be as follows: S0 to S1 to S2 to S3. If Clockwise is '0', the Stepper Motor will rotate counter-clockwise and the state change will be as follows: S0 to S3 to S2 to S1. Refer to Figure 26: MotorController Module Code for full view of code.

TimerReset Module

For the TimerReset Module, a Moore Machine will be used to determine when the timer reset will happen. Initially, the state will be at Car. When the state changes from Car to Train due to Train Button is pressed, Found will output '1'. This means that the timer will be reset. The same happens when Train state changes to Car or Pedestrian state and Pedestrian state changes to Train state. But when Car state changes to Pedestrian state, Found will output '0' which means the timer will not be reset. However, when Pedestrian state changes to Car state, Found will output '1' to reset the timer. Refer to Figure 29: TimerReset Module Code for full view of code.

Detailed Description of Simulation

SyncButton Module

TB_SyncButton.vhd Wed May 26 14:13:56 2021

```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   15:58:01 05/20/2021
6   -- Design Name:
7   -- Module Name:   C:/Xilinx/14.7/VHDL_Design_Project/TB_SyncButton.vhd
8   -- Project Name:  VHDL_Design_Project
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: SyncButton
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY TB_SyncButton IS
36  END TB_SyncButton;
37
38  ARCHITECTURE behavior OF TB_SyncButton IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT SyncButton
43      PORT(
44           reset : IN  std_logic;
45           clock : IN  std_logic;
46           TrainButton : IN  std_logic; -- Train Button Input --
47           CarButton : IN  std_logic; -- Car Button Input --
48           PedButton : IN  std_logic; -- Pedestrain Button Input --
49           Counter : IN  std_logic_vector(4 downto 0); -- Timer --
50           MotorEnabler : OUT  STD_LOGIC; -- Enable And Disable Stepper Motor --
51           TrainAction : OUT  std_logic; -- Tran Button Output --
52           CarWaitOutput : OUT  std_logic; -- Car Output When Train Is And Is Not
    Passing Through --
53           PedWaitOutput : OUT  std_logic -- Pedestrain Output When Train Is And Is Not
    Passing Through --
54          );
55      END COMPONENT;
```

Page 1

TB_SyncButton.vhd **Wed May 26 14:13:56 2021**

```
56
57
58      --Inputs
59      signal reset : std_logic := '0';
60      signal clock : std_logic := '0';
61      signal TrainButton : std_logic := '0';
62      signal CarButton : std_logic := '0';
63      signal PedButton : std_logic := '0';
64      signal Counter : std_logic_vector(4 downto 0) := (others => '0');
65
66      --Outputs
67      signal TrainAction : std_logic;
68      signal CarWaitOutput : std_logic;
69      signal PedWaitOutput : std_logic;
70      signal MotorEnabler : std_logic;
71
72      -- Clock period definitions
73      constant clock_period : time := 10 ns;
74
75  BEGIN
76
77      -- Instantiate the Unit Under Test (UUT)
78      uut: SyncButton PORT MAP (
79             reset => reset,
80             clock => clock,
81             TrainButton => TrainButton,
82             CarButton => CarButton,
83             PedButton => PedButton,
84             Counter => Counter,
85             TrainAction => TrainAction,
86             CarWaitOutput => CarWaitOutput,
87             PedWaitOutput => PedWaitOutput,
88             MotorEnabler => MotorEnabler
89           );
90
91      -- Clock process definitions
92      clock_process :process
93      begin
94         clock <= '0';
95         wait for clock_period/2;
96         clock <= '1';
97         wait for clock_period/2;
98      end process;
99
100
101     -- Stimulus process
102     stim_proc: process
103     begin
104
105        reset <= '1';
106
107        -- hold reset state for 100 ns.
108        wait for 100 ns;
109        reset <= '0';
110
111        -- insert stimulus here --
112        -- Only Train Button Pressed --
```

TB_SyncButton.vhd Wed May 26 14:13:56 2021

```
113         wait for clock_period*10;
114         TrainButton <= '1';
115         CarButton <= '0';
116         PedButton <= '0';
117         Counter <= "00000";
118
119         wait for clock_period*10;
120         TrainButton <= '0';
121         CarButton <= '0';
122         PedButton <= '0';
123         Counter <= "11101";
124
125         wait for clock_period*10;
126         TrainButton <= '0';
127         CarButton <= '0';
128         PedButton <= '0';
129         Counter <= "00101";
130
131         wait for clock_period*10;
132         TrainButton <= '0';
133         CarButton <= '0';
134         PedButton <= '0';
135         Counter <= "11110";
136
137         wait for clock_period*10;
138         TrainButton <= '0';
139         CarButton <= '0';
140         PedButton <= '0';
141         Counter <= "11110";
142
143         -- Car And Pedestrian Button Pressed When Train Passing --
144         wait for clock_period*10;
145         TrainButton <= '1';
146         CarButton <= '0';
147         PedButton <= '0';
148         Counter <= "00000";
149
150         wait for clock_period*10;
151         TrainButton <= '0';
152         CarButton <= '1';
153         PedButton <= '0';
154         Counter <= "11101";
155
156         wait for clock_period*10;
157         TrainButton <= '0';
158         CarButton <= '0';
159         PedButton <= '0';
160         Counter <= "11001";
161
162         wait for clock_period*10;
163         TrainButton <= '0';
164         CarButton <= '0';
165         PedButton <= '1';
166         Counter <= "10101";
167
168         wait for clock_period*10;
169         TrainButton <= '0';
```

Page 3

TB_SyncButton.vhd **Wed May 26 14:13:56 2021**

```
170         CarButton <= '0';
171         PedButton <= '0';
172         Counter <= "00101";
173
174         wait for clock_period*10;
175         TrainButton <= '0';
176         CarButton <= '0';
177         PedButton <= '0';
178         Counter <= "11110";
179
180         -- Car Button Pressed After Train Passed Through --
181         wait for clock_period*10;
182         TrainButton <= '0';
183         CarButton <= '1';
184         PedButton <= '0';
185         Counter <= "11000";
186
187         wait for clock_period*10;
188         TrainButton <= '0';
189         CarButton <= '0';
190         PedButton <= '0';
191         Counter <= "10000";
192
193         wait for clock_period*10;
194         TrainButton <= '0';
195         CarButton <= '0';
196         PedButton <= '0';
197         Counter <= "00101";
198
199         wait for clock_period*10;
200         TrainButton <= '0';
201         CarButton <= '0';
202         PedButton <= '0';
203         Counter <= "11000";
204
205         -- Pedestrain Button Pressed After Train Passed --
206         wait for clock_period*10;
207         TrainButton <= '1';
208         CarButton <= '0';
209         PedButton <= '0';
210         Counter <= "11110";
211
212         wait for clock_period*10;
213         TrainButton <= '0';
214         CarButton <= '0';
215         PedButton <= '1';
216         Counter <= "11100";
217
218         wait for clock_period*10;
219         TrainButton <= '0';
220         CarButton <= '0';
221         PedButton <= '0';
222         Counter <= "00100";
223
224         wait for clock_period*10;
225         TrainButton <= '0';
226         CarButton <= '0';
```

Page 4

TB_SyncButton.vhd Wed May 26 14:13:56 2021

```
227         PedButton <= '0';
228         Counter <= "00101";
229
230         wait for clock_period*10;
231         TrainButton <= '0';
232         CarButton <= '0';
233         PedButton <= '0';
234         Counter <= "11110";
235
236         -- Pedestrian Button Pressed When Gate Is Open --
237         wait for clock_period*10;
238         TrainButton <= '0';
239         CarButton <= '0';
240         PedButton <= '1';
241         Counter <= "00000";
242
243         wait for clock_period*10;
244         TrainButton <= '0';
245         CarButton <= '0';
246         PedButton <= '0';
247         Counter <= "11101";
248
249         wait for clock_period*10;
250         TrainButton <= '0';
251         CarButton <= '0';
252         PedButton <= '0';
253         Counter <= "00101";
254
255         wait for clock_period*10;
256         TrainButton <= '0';
257         CarButton <= '0';
258         PedButton <= '0';
259         Counter <= "10100";
260
261         -- Train Button Pressed When Pedestrian Is Passing --
262         wait for clock_period*10;
263         TrainButton <= '0';
264         CarButton <= '0';
265         PedButton <= '1';
266         Counter <= "00000";
267
268         wait for clock_period*10;
269         TrainButton <= '0';
270         CarButton <= '0';
271         PedButton <= '0';
272         Counter <= "00100";
273
274         wait for clock_period*10;
275         TrainButton <= '1';
276         CarButton <= '0';
277         PedButton <= '0';
278         Counter <= "00100";
279         wait;
280     end process;
281
282 END;
283
```

Page 5

Figure 12: TB_SyncButton TestBench Code

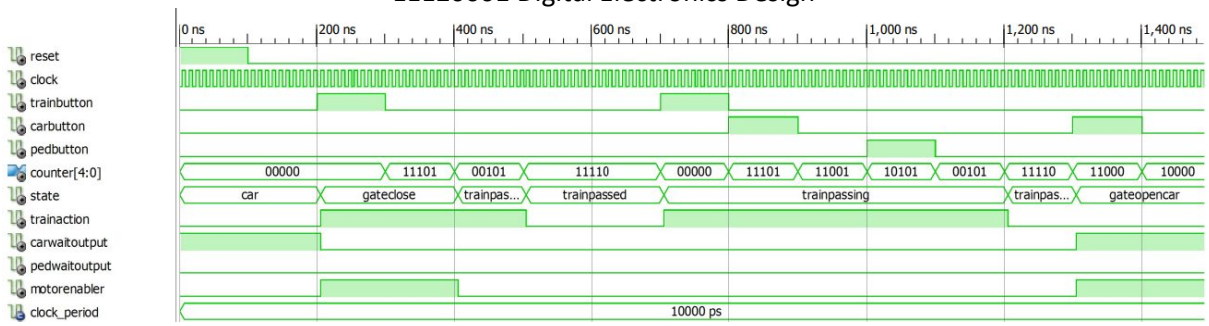

Figure 13: SyncButton Module Simulation at 0ns to 1500ns

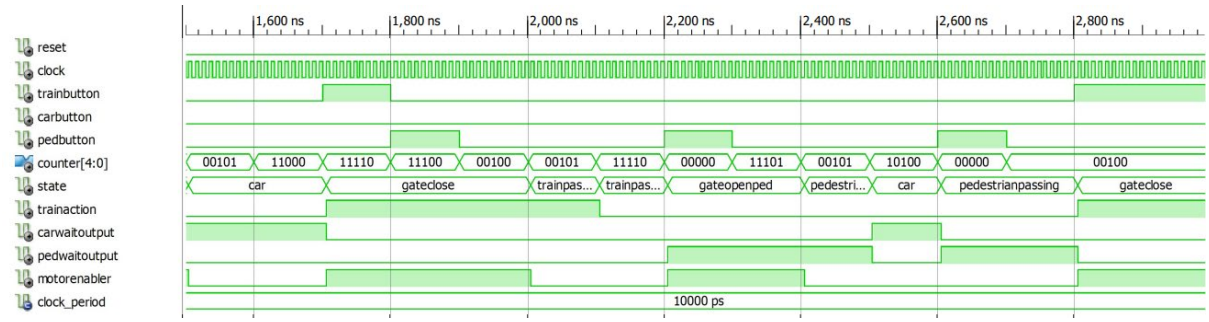

Figure 14: SyncButton Module Simulation at 1500ns to 3000ns

As shown in Figure 13: SyncButton Module Simulation at 0ns to 1500ns and Figure 14: SyncButton Module Simulation at 1500ns to 3000ns, the state will initially be at Car state. When the train button is pressed, it will go to GateClose state until Counter reaches "00101" then it will move to TrainPassing state. Then, when Counter reaches "11110", it will move to TrainPassed state. Note that during TrainPassing state, it will prioritize the train input even if the car and pedestrian button is pressed. When at TrainPassed state, if car button is pressed, it will move to GateOpenCar state. When Counter reaches "00101", it will move to Car state. If train button is pressed again, it will move to GateClose state and repeat the steps. Similarly, when the pedestrian button is pressed at TrainPassed state, the state will move to GateOpenPed state until "00101" where it will move to PedestrianPassing state until Counter reaches "10100" where it will move to Car state after "10100". But when the train button is pressed when the pedestrian is still passing in the PedestrianPassing state, the train will take first priority and the gate will state to close at GateClose state. Note that when at GateClose, GateOpenCar and GateOpenPed state, the MotorEnabler is '1', this is to enable the Stepper Motor to rotate. And also, during each specific state, different output will be '1'. For instance, at GateClose, TrainPassing and TrainPassed state, the TrainAction is '1'. At GateOpenCar and Car state, the CarWaitOutput is '1' and lastly at GateOpenPed and PedestrianPassing state, the PedWaitOutput is '1'.

TrafficLights Module

TB_TrafficLights.vhd **Mon May 24 17:06:00 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   12:41:02 05/20/2021
6   -- Design Name:
7   -- Module Name:   C:/Xilinx/14.7/VHDL_Design_Project/TB_TrafficLights.vhd
8   -- Project Name:  VHDL_Design_Project
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: TrafficLights
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  ----------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY TB_TrafficLights IS
36  END TB_TrafficLights;
37
38  ARCHITECTURE behavior OF TB_TrafficLights IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT TrafficLights
43      PORT(
44           reset : IN  std_logic;
45           clock : IN  std_logic;
46           FourHzPulse : IN  std_logic; -- 4Hz Pulse --
47           Flash : IN  std_logic; -- Train Button --
48           TrafficGreen : IN  std_logic; -- Car Button --
49           PedGreen : IN  std_logic; -- Pedestrian Button --
50           MotorEnable : OUT  std_logic; -- Enable The Stepper Motor To Rotate --
51           MotorClockwise : OUT  std_logic; -- Direction Of Rotation For Stepper Motor--
52           HTrafficLightOutput : OUT  std_logic_vector(1 downto 0); -- Output for
     Horizontal Traffic Light --
53           VTrafficLightOutput : OUT  std_logic_vector(1 downto 0) -- Output for
     Vertical Traffic Light --
54          );
55      END COMPONENT;
```

Page 1

TB_TrafficLights.vhd **Mon May 24 17:06:00 2021**

```
56
57
58      --Inputs
59      signal reset : std_logic := '0';
60      signal clock : std_logic := '0';
61      signal FourHzPulse : std_logic := '0';
62      signal Flash : std_logic := '0';
63      signal TrafficGreen : std_logic := '0';
64      signal PedGreen : std_logic := '0';
65
66      --Outputs
67      signal MotorEnable : std_logic;
68      signal MotorClockwise : std_logic;
69      signal HTrafficLightOutput : std_logic_vector(1 downto 0);
70      signal VTrafficLightOutput : std_logic_vector(1 downto 0);
71
72      -- Clock period definitions
73      constant clock_period : time := 10 ns;
74
75   BEGIN
76
77      -- Instantiate the Unit Under Test (UUT)
78      uut: TrafficLights PORT MAP (
79             reset => reset,
80             clock => clock,
81             FourHzPulse => FourHzPulse,
82             Flash => Flash,
83             TrafficGreen => TrafficGreen,
84             PedGreen => PedGreen,
85             MotorEnable => MotorEnable,
86             MotorClockwise => MotorClockwise,
87             HTrafficLightOutput => HTrafficLightOutput,
88             VTrafficLightOutput => VTrafficLightOutput
89           );
90
91      -- Clock process definitions
92      clock_process :process
93      begin
94         clock <= '0';
95         wait for clock_period/2;
96         clock <= '1';
97         wait for clock_period/2;
98      end process;
99
100     -- 4 Hz Pulse --
101     FourHzPulse_process :process
102     begin
103        FourHzPulse <= '0';
104        wait for 2.5 ns ;
105        FourHzPulse <= '1';
106        wait for clock_period/2;
107     end process;
108
109     -- Stimulus process
110     stim_proc: process
111     begin
112        -- hold reset state for 100 ns.
```

Page 2

TB_TrafficLights.vhd Mon May 24 17:06:00 2021

```
113         reset <= '1';
114         wait for 100 ns;
115         reset <= '0';
116
117         -- insert stimulus here --
118
119         -- When Train Button Is Pressed --
120         wait for clock_period*10;
121         Flash <= '1';
122         TrafficGreen <= '0';
123         PedGreen <= '0';
124
125         -- When Car Button Is Pressed --
126         wait for clock_period*10;
127         Flash <= '0';
128         TrafficGreen <= '1';
129         PedGreen <= '0';
130
131         -- When Pedestrian Button Is Pressed --
132         wait for clock_period*10;
133         Flash <= '0';
134         TrafficGreen <= '0';
135         PedGreen <= '1';
136
137         -- When Train Button Is Pressed --
138         wait for clock_period*10;
139         Flash <= '1';
140         TrafficGreen <= '0';
141         PedGreen <= '0';
142
143         wait;
144     end process;
145
146 END;
```

Page 3

Figure 15: TB_TrafficLights TestBench Code

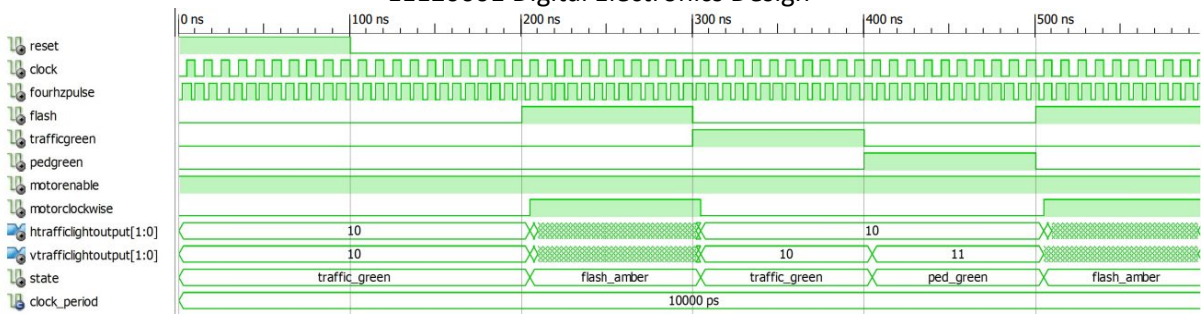

Figure 16: TrafficLigths Module Simulation

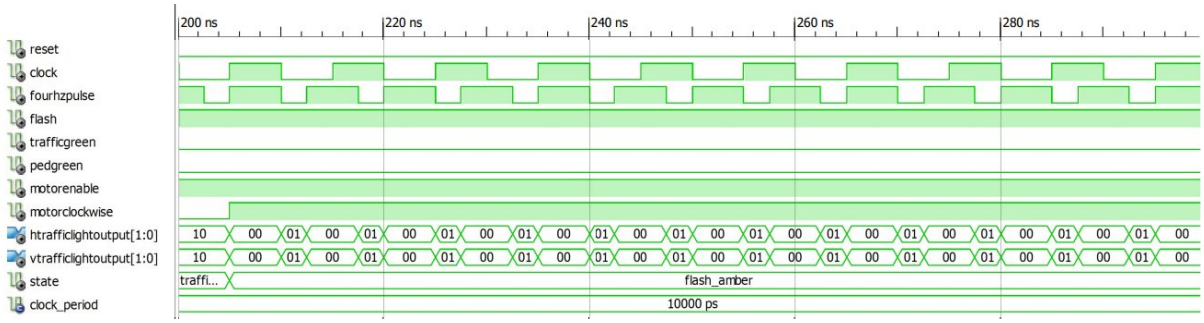

Figure 17: TrafficLights Module Simulation at 200ns to 300ns

As shown in Figure 17: TrafficLights Module Simulation at 200ns to 300ns, the initial state will be Traffic_Green state. When Flash is '1', meaning train button is pressed, the state will move to Flash_Amber state where the output for both traffic lights will be "00" and "01". The MotorClockwise will also output '1' to make the Stepper Motor rotate in a clockwise direction. When the car button is pressed, the state will move to Traffic_Green state where both traffic lights will output "10". Then, when the pedestrian button is pressed, the state will move to Ped_Green state where both traffic lights will turn green, and the pedestrian light will be on indicated by the "11" output from VTrafficLightOutput. In both Traffic_Green and Ped_Green state, the MotorClockwise output will be '0' to make the Stepper Motor rotate in a counter-clockwise direction. In all the states, the MotorEnable output will be '1' to enable to Stepper Motor. Refer to FIG XX for a full view of the testbench code.

Timer Module

TB_Timer.vhd **Sat May 22 00:25:42 2021**

```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   15:40:42 05/15/2021
6   -- Design Name:
7   -- Module Name:   C:/Xilinx/14.7/VHDL_Design_Project/TB_Timer.vhd
8   -- Project Name:  VHDL_Design_Project
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: Timer
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY TB_Timer IS
36  END TB_Timer;
37
38  ARCHITECTURE behavior OF TB_Timer IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT Timer
43      PORT(
44           reset : IN  std_logic;
45           clock : IN  std_logic;
46           timer_reset : IN  std_logic; -- To Reset Timer --
47           FourHzPulse : OUT  std_logic; -- Four Hz Pulse --
48           MotorPulseOutput : OUT  std_logic; -- Pulse For Stepper Motor --
49           CounterOutput : OUT  std_logic_vector(4 downto 0) -- Timer Output --
50          );
51      END COMPONENT;
52
53
54     --Inputs
55     signal reset : std_logic := '0';
56     signal clock : std_logic := '0';
57     signal timer_reset : std_logic := '0';
```

Page 1

TB_Timer.vhd Sat May 22 00:25:43 2021

```
58
59      --Outputs
60      signal FourHzPulse : std_logic;
61      signal MotorPulseOutput : std_logic;
62      signal CounterOutput : std_logic_vector(4 downto 0);
63
64      -- Clock period definitions
65      constant clock_period : time := 10 us;
66
67   BEGIN
68
69      -- Instantiate the Unit Under Test (UUT)
70      uut: Timer PORT MAP (
71             reset => reset,
72             clock => clock,
73             timer_reset => timer_reset,
74             FourHzPulse => FourHzPulse,
75             MotorPulseOutput => MotorPulseOutput,
76             CounterOutput => CounterOutput
77           );
78
79      -- Clock process definitions
80      clock_process :process
81      begin
82         clock <= '0';
83         wait for clock_period/2;
84         clock <= '1';
85         wait for clock_period/2;
86      end process;
87
88
89      -- Stimulus process
90      stim_proc: process
91      begin
92         -- hold reset state for 100 ns.
93         reset <= '1';
94
95         wait for 100 ns;
96         reset <= '0';
97
98         -- insert stimulus here --
99         -- If timer_reset Is 0, Meaning Timer Will Not Be Reset --
100        timer_reset <= '0';
101        wait for clock_period*10000;
102
103        -- If timer_reset Is 1, Meaning Timer Will Be Reset --
104        timer_reset <= '1';
105
106        wait;
107     end process;
108
109  END;
110
```

Page 2

Figure 18: TB_Timer TestBench Code

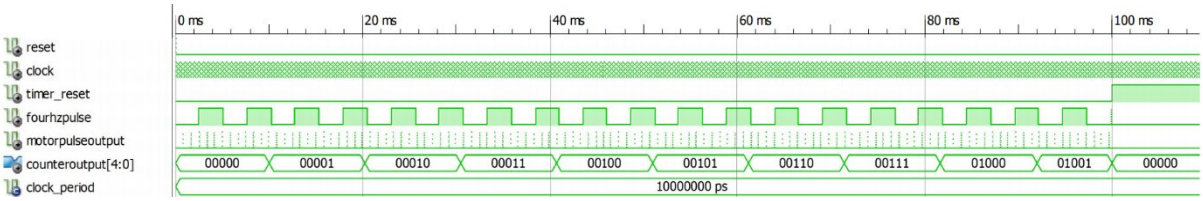

Figure 19: Timer Module Simulation

As shown in Figure 19: Timer Module Simulation, when the timer_reset is '1', the CounterOutput will reset back to "0000" and the MotorPulseOutput and FourHzPulse will be '0'. The CounterOutput will continue to count when the timer_reset is '0' following the positive clock edge and the FourHzPulse and MotorPulseOutput will also continue to pulse at set durations again. Refer to FID XX for full view of testbench code.

MotorController Module

TB_MotorController.vhd **Sat May 22 00:26:44 2021**

```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   22:05:53 05/14/2021
6   -- Design Name:
7   -- Module Name:   C:/Xilinx/14.7/VHDL_Design_Project/TB_MotorController.vhd
8   -- Project Name:  VHDL_Design_Project
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: MotorController
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY TB_MotorController IS
36  END TB_MotorController;
37
38  ARCHITECTURE behavior OF TB_MotorController IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT MotorController
43      PORT(
44           reset : IN  std_logic;
45           MotorPulseInput : IN  std_logic;
46           Enable : IN  std_logic; -- To Enable The Motor To Turn --
47           Clockwise : IN  std_logic; -- To Choose Direction To Turn The Stepper Motor
48           MotorEnablerInput : IN std_logic; -- To Enable And Disable The Stepper Motor
    --
49           MotorOutput : OUT  std_logic_vector(3 downto 0) -- Output for Motor --
50          );
51      END COMPONENT;
52
53
54     --Inputs
55     signal reset : std_logic := '0';
56     signal MotorPulseInput : std_logic := '0';
```

Page 1

TB_MotorController.vhd Sat May 22 00:26:44 2021

```
57      signal Enable : std_logic := '0';
58      signal Clockwise : std_logic := '0';
59      signal MotorEnablerInput : std_logic := '0';
60
61      --Outputs
62      signal MotorOutput : std_logic_vector(3 downto 0);
63
64      -- Clock period definitions
65      constant MotorPulseInput_period : time := 10 ns;
66
67  BEGIN
68
69      -- Instantiate the Unit Under Test (UUT)
70      uut: MotorController PORT MAP (
71             reset => reset,
72             MotorPulseInput => MotorPulseInput,
73             Enable => Enable,
74             Clockwise => Clockwise,
75             MotorEnablerInput => MotorEnablerInput,
76             MotorOutput => MotorOutput
77           );
78
79      -- Clock process definitions
80      MotorPulseInput_process :process -- Follows The Motor Pulse --
81      begin
82         MotorPulseInput <= '0';
83         wait for MotorPulseInput_period/2;
84         MotorPulseInput <= '1';
85         wait for MotorPulseInput_period/2;
86      end process;
87
88
89      -- Stimulus process
90      stim_proc: process
91      begin
92         -- hold reset state for 100 ns.
93         reset <= '1';
94
95         wait for 100 ns;
96         reset <= '0';
97
98         -- insert stimulus here --
99
100        -- When MotorEnablerInput Is 0, Meaning Stepper Motor Will Not Be Enabled --
101        wait for MotorPulseInput_period*10;
102        MotorEnablerInput <= '0';
103        Enable <= '0';
104        Clockwise <= '0';
105
106        wait for MotorPulseInput_period*10;
107        MotorEnablerInput <= '0';
108        Enable <= '1';
109        Clockwise <= '0';
110
111        wait for MotorPulseInput_period*10;
112        MotorEnablerInput <= '0';
113        Enable <= '0';
```

Page 2

TB_MotorController.vhd **Sat May 22 00:26:44 2021**

```
114         Clockwise <= '1';
115
116         wait for MotorPulseInput_period*10;
117         MotorEnablerInput <= '0';
118         Enable <= '1';
119         Clockwise <= '1';
120
121         -- When MotorEnablerInput Is 1, Meaning Stepper Motor Will Be Enabled --
122         wait for MotorPulseInput_period*10;
123         MotorEnablerInput <= '1';
124         Enable <= '0';
125         Clockwise <= '0';
126
127         wait for MotorPulseInput_period*10;
128         MotorEnablerInput <= '1';
129         Enable <= '1';
130         Clockwise <= '0';
131
132         wait for MotorPulseInput_period*10;
133         MotorEnablerInput <= '1';
134         Enable <= '0';
135         Clockwise <= '1';
136
137         wait for MotorPulseInput_period*10;
138         MotorEnablerInput <= '1';
139         Enable <= '1';
140         Clockwise <= '1';
141
142         wait;
143      end process;
144
145   END;
146
```

Page 3

Figure 20: TB_MotorController TestBench Code

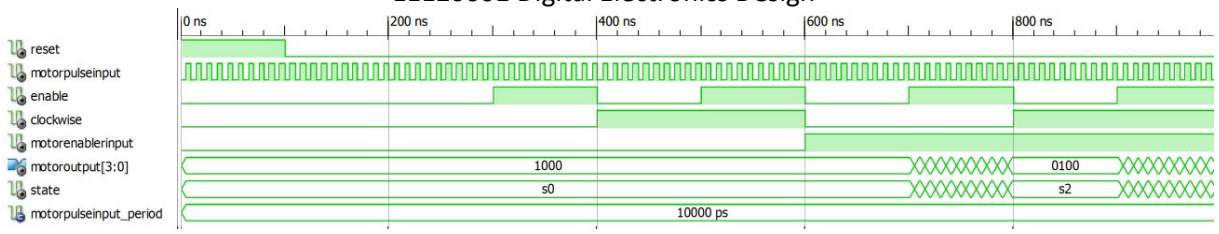

Figure 21: MotorController Module Simulation

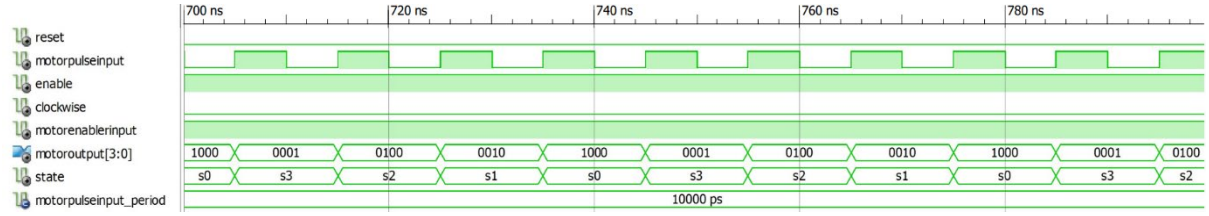

Figure 22: MotorController Module Simulation at 700ns to 800ns

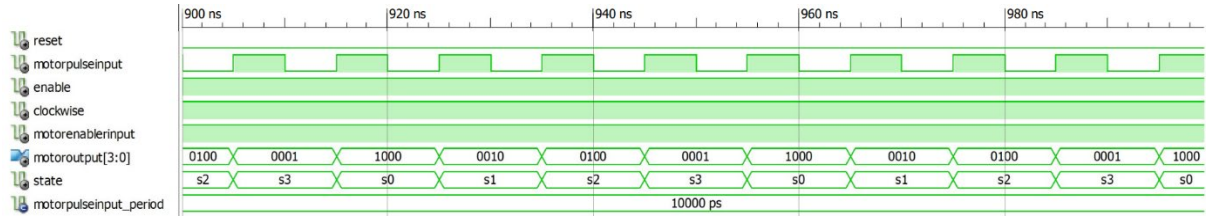

Figure 23: MotorController Module Simulation at 900ns to 1000ns

As shown in Figure 21: MotorController Module Simulation, the state will only change when both MotorEnablerInput and Enable are '1' and it will only changes when it detects MotorPulseInput is '1'. From Figure 22: MotorController Module Simulation at 700ns to 800ns, observing the state it can be seen that the state is decrementing or moving backwards. This means that the Stepper Motor will rotate in a counter-clockwise direction thus the clockwise input is '0'. From Figure 23: MotorController Module Simulation at 900ns to 1000ns, observing the state, the state is incrementing or moving forward. This means that the Stepper Motor will rotate in a clockwise direction thus the clockwise input is '1'. One thing to note is that the state will only change when the MotorPulseInput is '1'.

TimerReset Module

TB_TimerReset.vhd **Wed May 26 11:31:16 2021**

```
1   --------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:   22:47:17 05/19/2021
6   -- Design Name:
7   -- Module Name:   C:/Xilinx/14.7/VHDL_Design_Project/TB_TimerReset.vhd
8   -- Project Name:  VHDL_Design_Project
9   -- Target Device:
10  -- Tool versions:
11  -- Description:
12  --
13  -- VHDL Test Bench Created by ISE for module: TimerReset
14  --
15  -- Dependencies:
16  --
17  -- Revision:
18  -- Revision 0.01 - File Created
19  -- Additional Comments:
20  --
21  -- Notes:
22  -- This testbench has been automatically generated using types std_logic and
23  -- std_logic_vector for the ports of the unit under test.  Xilinx recommends
24  -- that these types always be used for the top-level I/O of a design in order
25  -- to guarantee that the testbench will bind correctly to the post-implementation
26  -- simulation model.
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY TB_TimerReset IS
36  END TB_TimerReset;
37
38  ARCHITECTURE behavior OF TB_TimerReset IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT TimerReset
43      PORT(
44           reset : IN  std_logic;
45           clock : IN  std_logic;
46           TrainButton : IN  std_logic; -- Train Button Input --
47           CarButton : IN  std_logic; -- Car Button Input --
48           PedButton : IN  std_logic; -- Pedestrain Button Input --
49           Found : OUT  std_logic -- Output To Reset Timer --
50          );
51      END COMPONENT;
52
53
54     --Inputs
55     signal reset : std_logic := '0';
56     signal clock : std_logic := '0';
57     signal TrainButton : std_logic := '0';
```

Page 1

TB_TimerReset.vhd **Wed May 26 11:31:16 2021**

```
58      signal CarButton : std_logic := '0';
59      signal PedButton : std_logic := '0';
60
61      --Outputs
62      signal Found : std_logic;
63
64      -- Clock period definitions
65      constant clock_period : time := 10 ns;
66
67  BEGIN
68
69      -- Instantiate the Unit Under Test (UUT)
70      uut: TimerReset PORT MAP (
71             reset => reset,
72             clock => clock,
73             TrainButton => TrainButton,
74             CarButton => CarButton,
75             PedButton => PedButton,
76             Found => Found
77           );
78
79      -- Clock process definitions
80      clock_process :process
81      begin
82         clock <= '0';
83         wait for clock_period/2;
84         clock <= '1';
85         wait for clock_period/2;
86      end process;
87
88
89      -- Stimulus process
90      stim_proc: process
91      begin
92
93         reset <= '1';
94
95         -- hold reset state for 100 ns.
96         wait for 100 ns;
97         reset <= '0';
98
99         -- insert stimulus here --
100
101        -- Observe When The Found Output Is 1 --
102        wait for clock_period*10;
103        TrainButton <= '1';
104        CarButton <= '0';
105        PedButton <= '0';
106
107        wait for clock_period*10;
108        TrainButton <= '0';
109        CarButton <= '1';
110        PedButton <= '0';
111
112        wait for clock_period*10;
113        TrainButton <= '1';
114        CarButton <= '0';
```

Page 2

TB_TimerReset.vhd Wed May 26 11:31:16 2021

```
115         PedButton <= '0';
116
117         wait for clock_period*10;
118         TrainButton <= '0';
119         CarButton <= '0';
120         PedButton <= '1';
121
122         wait for clock_period*10;
123         TrainButton <= '0';
124         CarButton <= '1';
125         PedButton <= '0';
126
127         wait for clock_period*10;
128         TrainButton <= '0';
129         CarButton <= '0';
130         PedButton <= '1';
131
132         wait for clock_period*10;
133         TrainButton <= '1';
134         CarButton <= '0';
135         PedButton <= '0';
136
137         wait;
138     end process;
139
140 END;
141
```

Page 3

Figure 24: TB_ TimerReset TestBench Code

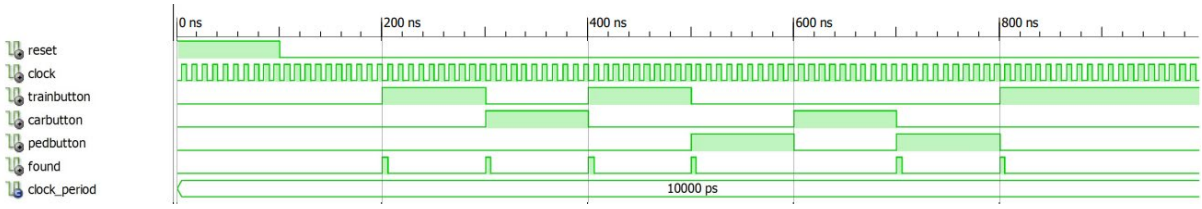

Figure 25: TimerReset Module Simulation

As shown in Figure 25: TimerReset Module Simulation, when Car state is changed to Train state or vice versa, the Found output will be ‘1’ to reset the timer. The same happens when Train state moved to Pedestrian state and vice versa. But when Pedestrian state moved to Car state or vice versa, the Found output will be ‘0’ because the timer does not need to be reset. But when Car state moved to Pedestrian state, the Found output will be ‘1’.

Detailed Description on Debugging Process

Stepper Motor Will Not Stop Rotating After 5 rotations

It was found that the stepper motor will continue to rotate even after 5 rotations due to our code not disabling the stepper motor after it has rotated 5 times. To fix this, we have added 3 more states in the SyncButton Module for when the gate is closing, gate opening when pedestrian button is pressed and gate opening when car button is pressed after the train has passed.

```
--
            case state is

               when GateClose  => -- When The State Is GateClose --
                  TrainAction <= '1'; -- Set TrainAction Output To Be 1 --
                  CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
                  PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
                  MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --

                  if (Counter = "00101") then -- If Counter Reached "00101" --
                     NextState <= TrainPassing; -- Change State To TrainPassing --

                  else
                     NextState <= GateClose; -- If None Of The Above Occur, Hold State --

                  end if;

               when GateOpenCar  => -- When The State Is GateOpenCar --
                  TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
                  CarWaitOutput <= '1'; -- Set CarWaitOutput Output To Be 1 --
                  PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
                  MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --

                  if (Counter = "00101") then -- If Counter Reached "00101" --
                     NextState <= Car; -- Change State To Car --

                  else
                     NextState <= GateOpenCar; -- If None Of The Above Occur, Hold State
      --

                  end if;
```

SyncButton.vhd Sat May 22 00:27:07 2021

```
               when GateOpenPed  => -- When The State Is GateOpenPed --
                  TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
                  CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
                  PedWaitOutput <= '1'; -- Set PedWaitOutput Output To Be 1 --
                  MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --

                  if (Counter = "00101") then -- If Counter Reached "00101" --
                     NextState <= PedestrianPassing; -- Change State To Car --

                  else
                     NextState <= GateOpenPed; -- If None Of The Above Occur, Hold State
      --

                  end if;
```

The MotorEnabler output is to control when to enable the stepper motor during these states, the MotorEnabler output will be 0 when it has exited these three states meaning that the gate has completely closed or opened when the Counter has reached “00101” or 5 seconds. Refer to Figure 26: MotorController Module Code for the full view of the code.

Traffic Light Flashing Is Not Consistent and Accurate

It was found that the traffic light would flash red and amber with glitches. It will flash with an inconsistent timing due to the states that are created in our previous coding in TrafficLights Module. Previously, we have implemented two states for the traffic lights to flash, which is Flash_Amber state and Flash_Red states. The inconsistent timing only occurred when we have downloaded the code onto the CMOD chips and observed it on the hardware, surprisingly it was glitch free during the simulation. Because of the two states switching back and forth when a 4Hz pulse is detected, it caused the inconsistency to occur. To fix this, we removed one of the states, as for our current code, we have removed the Flash_Red state. Refer to Figure 28: TrafficLights Module Code for the full view of the code. After removing one of the states and making the output change from red to amber when it detects a 4Hz pulse, it has solved the inconsistency.

```
85              when Flash_Amber  => -- When The State Is Flash_Amber --
86                 MotorEnable <= '1'; -- Stepper Motor Is Enabled --
87                 MotorClockwise <= '1'; -- Stepper Motor Rotate Clockwise --
88
89                 if (FourHzPulse = '1') then -- If FourHzPulse Output Is 1, Meaning It
    Reached 4 Hz --
90                    HTrafficLightOutput <= "00"; -- Set Traffic Light To Red --
91                    VTrafficLightOutput <= "00"; -- Set Traffic Light To Red --
92                 else
93                    HTrafficLightOutput <= "01"; -- Set Traffic Light To Amber --
94                    VTrafficLightOutput <= "01"; -- Set Traffic Light To Amber --
95
96                 end if;
97
98                 if (PedGreen = '1') then -- If Pedestrian Button Is Pressed --
99                    NextState <= Ped_Green; -- Change State to Ped_Green --
100
101                elsif (TrafficGreen = '1') then -- If Car Button Is Pressed --
102                   NextState <= Traffic_Green; -- Change State to Traffic_Green --
103
104                else
105                   NextState <= Flash_Amber; -- If None Of The Above Occur, Hold State
    --
106
107                end if;
108
```

Listing of VHDL Modules

MotorController.vhd **Sat May 22 00:26:29 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    21:49:37 05/14/2021
6   -- Design Name:
7   -- Module Name:    MotorController - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity MotorController is
33      Port ( reset : in  STD_LOGIC;
34             MotorPulseInput : in  STD_LOGIC;
35             Enable : in  STD_LOGIC; -- To Enable The Motor To Turn --
36             Clockwise : in  STD_LOGIC; -- To Choose Direction To Turn The Stepper
    Motor --
37             MotorEnablerInput : in STD_LOGIC; -- To Enable And Disable The Stepper
    Motor --
38             MotorOutput : out  STD_LOGIC_VECTOR (3 downto 0) -- Output for Motor --
39             );
40  end MotorController;
41
42  architecture Behavioral of MotorController is
43
44  type StateType is (S0, S1, S2, S3);
45  -- S0 = '1000' as Motor Output --
46  -- S1 = '0010' as Motor Output --
47  -- S2 = '0100' as Motor Output --
48  -- S3 = '0001' as Motor Output --
49
50  signal State, NextState : StateType; -- Moore Machine
51
52  begin
53
54     SyncProcess:
55        process (reset, MotorPulseInput)
```

Page 1

MotorController.vhd **Sat May 22 00:26:30 2021**

```
56
57          begin
58             if (reset = '1') then -- If reset Button Is Pressed --
59                State <= S0; -- Change The State Back To S0 --
60
61             elsif (rising_edge(MotorPulseInput)) then -- If Positive Clock Edge of
    MotorPulse --
62                State <= NextState; -- Change The State To The Next State --
63
64             end if;
65
66          end process;
67
68       MotorCombinationProcess:
69          process (State, MotorEnablerInput, Enable, Clockwise)
70
71          begin
72
73             MotorOutput <= "1000"; -- Set Motor Output To '1000' --
74             NextState <= S0; -- Set The State To S0 --
75
76             case State is
77
78                when S0 => -- When State Is At S0 --
79                   MotorOutput <= "1000"; -- Set Motor Output To '1000' --
80
81                   if (MotorEnablerInput = '1') then -- Is MotorEnablerInput Is 1 --
82
83                      if (Enable = '1') then -- If Enable Is 1 --
84
85                         if (Clockwise = '1') then -- If Clockwise Is 1 --
86                            NextState <= S1; -- Change The State To The S1 --
87
88                         else -- If Clockwise Is 0 --
89                            NextState <= S3; -- Change The State To The S3 --
90
91                         end if;
92
93                      else -- If Enable Output Is 0 --
94                         NextState <= S0; -- Hold The Current State --
95
96                      end if;
97
98                   else -- Is MotorEnablerInput Is 0 --
99                      NextState <= S0; -- Hold The Current State --
100
101                  end if;
102
103               when S1 => -- When State Is At S1 --
104                  MotorOutput <= "0010"; -- Set Motor Output To '0010' --
105
106                  if (MotorEnablerInput = '1') then -- Is MotorEnablerInput Is 1 --
107
108                     if (Enable = '1') then -- If Enable Is 1 --
109
110                        if (Clockwise = '1') then -- If Clockwise Is 1 --
111                           NextState <= S2; -- Change The State To The S2 --
```

Page 2

MotorController.vhd Sat May 22 00:26:30 2021

```
112
113                     else -- If Clockwise Is 0 --
114                        NextState <= S0; -- Change The State To The S0 --
115
116                     end if;
117
118                  else -- If Enable Is 0 --
119                     NextState <= S1; -- Hold The Current State --
120
121                  end if;
122
123               else -- Is MotorEnablerInput Is 0 --
124                  NextState <= S1; -- Hold The Current State --
125
126               end if;
127
128            when S2 => -- When State Is At S2 --
129               MotorOutput <= "0100"; -- Set Motor Output To '0100' --
130
131               if (MotorEnablerInput = '1') then -- Is MotorEnablerInput Is 1 --
132
133                  if (Enable = '1') then -- If Enable Is 1 --
134
135                     if (Clockwise = '1') then -- If Clockwise Is 1 --
136                        NextState <= S3; -- Change The State To The S3 --
137
138                     else -- If Clockwise Is 0 --
139                        NextState <= S1; -- Change The State To The S1 --
140
141                     end if;
142
143                  else -- If Enable Is 0 --
144                     NextState <= S2; -- Hold The Current State --
145
146                  end if;
147
148               else -- Is MotorEnablerInput Is 0 --
149                  NextState <= S2; -- Hold The Current State --
150
151               end if;
152
153            when S3 => -- When State Is At S3 --
154               MotorOutput <= "0001"; -- Set Motor Output To '0001' --
155
156               if (MotorEnablerInput = '1') then -- Is MotorEnablerInput Is 1 --
157
158                  if (Enable = '1') then -- If Enable Is 1 --
159
160                     if (Clockwise = '1') then -- If Clockwise Is 1 --
161                        NextState <= S0; -- Change The State To The S0 --
162
163                     else -- If Clockwise Is 0 --
164                        NextState <= S2; -- Change The State To The S2 --
165
166                     end if;
167
168                  else -- If Enable Is 0 --
```

Page 3

MotorController.vhd Sat May 22 00:26:30 2021

```
169                         NextState <= S3; -- Hold The Current State --
170
171                      end if;
172
173                   else -- Is MotorEnablerInput Is 0 --
174                      NextState <= S3; -- Hold The Current State --
175
176                   end if;
177
178                end case;
179
180          end process;
181
182    end Behavioral;
```

Page 4

Figure 26: MotorController Module Code

Timer.vhd **Sat May 22 00:25:28 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    14:52:33 05/14/2021
6   -- Design Name:
7   -- Module Name:    TimerMealyMachine - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23
24  -- Uncomment the following library declaration if using
25  -- arithmetic functions with Signed or Unsigned values
26
27
28  -- Uncomment the following library declaration if instantiating
29  -- any Xilinx primitives in this code.
30  --library UNISIM;
31  --use UNISIM.VComponents.all;
32
33  entity Timer is
34      Port ( reset : in  STD_LOGIC;
35             clock : in  STD_LOGIC;
36             timer_reset : in  STD_LOGIC; -- To Reset Timer --
37             FourHzPulse : out  STD_LOGIC; -- Four Hz Pulse --
38             MotorPulseOutput : out STD_LOGIC; -- Pulse For Stepper Motor --
39             CounterOutput : out  STD_LOGIC_VECTOR (4 downto 0) -- Timer Output --
40             );
41  end Timer;
42
43  architecture Behavioral of Timer is
44
45  -- 1KHz = 1000 counts per second --
46  -- 1 second = 1000 counts --
47  -- 0.25 second = 250 counts --
48  -- 1 rotation in 1 second = 12 pulses = 83 counts --
49
50  signal counter : unsigned (9 downto 0); -- Unsigned Signal To Store Counter for
    Generating Pulses --
51  signal timekeeper : unsigned (4 downto 0); -- Unsigned Signal To Store The Amount Of
    Seconds --
52
53  begin
54
55  CounterOutput <= STD_LOGIC_VECTOR (timekeeper); -- Set CounterOutput Equal To
```

Page 1

Timer.vhd Sat May 22 00:25:29 2021

```
    timekeeper --
56  FourHzPulse <= counter (8); -- Generate 4Hz Pulse According To The 8th Digit Of
    Counter --
57
58
59     process (reset, timer_reset, clock) -- Whenever A Button Is Pressed, It Will Reset
    The Timer; Button Is Mapped To Reset --
60
61     variable MotorCount: unsigned (9 downto 0); -- Unsigned Variable To Keep Track Of
    Motor Pulse --
62
63     begin
64        if (reset = '1') then -- Master Reset --
65           counter <= (others => '0'); -- Reset counter To 0 --
66           timekeeper <= (others => '0'); -- Reset timekeeper To 0 --
67           MotorPulseOutput <= '0'; -- Reset MotorPulse To 0 --
68           MotorCount := "0001010011"; -- First Motor Pulse --
69
70        elsif (timer_reset = '1') then -- If Timer Is To Be Reset --
71           counter <= (others => '0'); -- Reset counter To 0 --
72           timekeeper <= (others => '0'); -- Reset timekeeper To 0 --
73           MotorPulseOutput <= '0'; -- Reset MotorPulse To 0 --
74           MotorCount := "0001010011"; -- First Motor Pulse --
75
76        elsif (rising_edge(clock)) then
77           counter <= counter + 1; -- Increment counter By 1 --
78
79           if (counter = "1111101000") then -- If counter Reached 1 second --
80              timekeeper <= timekeeper + 1; -- Increment timekeeper By 1 --
81
82           end if;
83
84           if (counter = MotorCount) then -- If counter Is The Same As MotorCount --
85              MotorPulseOutput <= '1'; -- MotorPulse Will Output 1 --
86              MotorCount := MotorCount + "0001010011"; -- Increment MotorPulse By 83
    counts --
87
88           else
89              MotorPulseOutput <= '0'; -- If counter Is Not The Same As MotorCount,
    MotorPulse Will Output 0 --
90
91           end if;
92
93        end if;
94
95     end process;
96  end Behavioral;
```

Page 2

Figure 27: Timer Module Code

TrafficLights.vhd **Mon May 24 17:05:37 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    14:32:48 05/14/2021
6   -- Design Name:
7   -- Module Name:    TrafficLights - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity TrafficLights is
33      Port ( reset : in  STD_LOGIC;
34             clock : in  STD_LOGIC;
35             FourHzPulse : in  STD_LOGIC; -- 4Hz Pulse --
36             Flash : in  STD_LOGIC; -- Train Button --
37             TrafficGreen : in  STD_LOGIC; -- Car Button --
38             PedGreen : in  STD_LOGIC; -- Pedestrian Button --
39             MotorEnable : out  STD_LOGIC; -- Enable The Stepper Motor To Rotate --
40             MotorClockwise : out  STD_LOGIC; -- Direction Of Rotation For Stepper
   Motor--
41             HTrafficLightOutput : out  STD_LOGIC_VECTOR (1 downto 0); -- Output for
   Horizontal Traffic Light --
42             VTrafficLightOutput : out  STD_LOGIC_VECTOR (1 downto 0) -- Output for
   Vertical Traffic Light --
43             );
44
45  end TrafficLights;
46
47  architecture Behavioral of TrafficLights is
48
49  type StateType is (Flash_Amber, Traffic_Green, Ped_Green);
50  -- Flash_Amber = Traffic Light Will Turn Flash Between Amber And Red --
51  -- Traffic_Green = Traffic Light Will Turn Green --
52  -- Ped_Green = Pedestrian Light Will Turn On And Traffic Light Will Turn Green --
53
54  signal State, NextState : StateType;
```

Page 1

TrafficLights.vhd **Mon May 24 17:05:37 2021**

```
 55
 56   begin
 57
 58      SyncProcess:
 59         process (reset, clock)
 60
 61         begin
 62            if (reset = '1') then -- If reset is pressed --
 63               State <= Traffic_Green; -- Change The State To Traffic_Green --
 64
 65            elsif (rising_edge(clock)) then -- If Rising Clock Edge --
 66               State <= NextState; -- Change The State To The Next State --
 67
 68            end if;
 69
 70         end process;
 71
 72
 73      MotorCombinationProcess:
 74         process (State, Flash, FourHzPulse, PedGreen, TrafficGreen)
 75
 76         begin
 77
 78            HTrafficLightOutput <= "10"; -- Set Traffic Light To Green --
 79            VTrafficLightOutput <= "10"; -- Set Traffic Light To Green --
 80
 81            NextState <= Traffic_Green; -- Set Current State To Traffic_Green --
 82
 83            case State is
 84
 85               when Flash_Amber  => -- When The State Is Flash_Amber --
 86                  MotorEnable <= '1'; -- Stepper Motor Is Enabled --
 87                  MotorClockwise <= '1'; -- Stepper Motor Rotate Clockwise --
 88
 89                  if (FourHzPulse = '1') then -- If FourHzPulse Output Is 1, Meaning It
    Reached 4 Hz --
 90                     HTrafficLightOutput <= "00"; -- Set Traffic Light To Red --
 91                     VTrafficLightOutput <= "00"; -- Set Traffic Light To Red --
 92                  else
 93                     HTrafficLightOutput <= "01"; -- Set Traffic Light To Amber --
 94                     VTrafficLightOutput <= "01"; -- Set Traffic Light To Amber --
 95
 96                  end if;
 97
 98                  if (PedGreen = '1') then -- If Pedestrian Button Is Pressed --
 99                     NextState <= Ped_Green; -- Change State to Ped_Green --
100
101                  elsif (TrafficGreen = '1') then -- If Car Button Is Pressed --
102                     NextState <= Traffic_Green; -- Change State to Traffic_Green --
103
104                  else
105                     NextState <= Flash_Amber; -- If None Of The Above Occur, Hold State
    --
106
107                  end if;
108
109               when Traffic_Green => -- When The State is Traffic_Green --
```

Page 2

TrafficLights.vhd Mon May 24 17:05:38 2021

```
110                 HTrafficLightOutput <= "10"; -- Set Traffic Light To Green --
111                 VTrafficLightOutput <= "10"; -- Set Traffic Light To Green --
112                 MotorEnable <= '1'; -- Stepper Motor Is Enabled --
113                 MotorClockwise <= '0'; -- Stepper Motor Rotate Counter-Clockwise --
114
115                 if (Flash = '1') then -- If Train Button Is Pressed --
116                    NextState <= Flash_Amber; -- Change State to Flash_Amber --
117
118                 elsif (PedGreen = '1') then -- If Pedestrian Button Is Pressed --
119                    NextState <= Ped_Green; -- Change State to Ped_Green --
120
121                 else
122                    NextState <= Traffic_Green; -- If None Of The Above Occur, Hold
    State --
123                 end if;
124
125              when Ped_Green => -- When The State is Ped_Green --
126                 HTrafficLightOutput <= "10"; -- Set Traffic Light To Green --
127                 VTrafficLightOutput <= "11"; -- Set Traffic Light & Pedestrain Light
    To Green --
128                 MotorEnable <= '1'; -- Stepper Motor Is Enabled --
129                 MotorClockwise <= '0'; -- Stepper Motor Rotate Counter-Clockwise --
130
131                 if (Flash = '1') then -- If Train Button Is Pressed --
132                    NextState <= Flash_Amber; -- Change State to Flash_Amber --
133
134                 elsif (TrafficGreen = '1') then -- If Car Button Is Pressed --
135                    NextState <= Traffic_Green; -- Change State to Traffic_Green --
136
137                 else
138                    NextState <= Ped_Green; -- If None Of The Above Occur, Hold State --
139                 end if;
140
141              end case;
142
143        end process;
144
145  end Behavioral;
```

Page 3

Figure 28: TrafficLights Module Code

TimerReset.vhd **Tue Jun 01 12:52:29 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    14:52:33 05/14/2021
6   -- Design Name:
7   -- Module Name:    TimerReset - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23
24  -- Uncomment the following library declaration if using
25  -- arithmetic functions with Signed or Unsigned values
26
27
28  -- Uncomment the following library declaration if instantiating
29  -- any Xilinx primitives in this code.
30  --library UNISIM;
31  --use UNISIM.VComponents.all;
32
33  entity TimerReset is
34      Port ( reset : in  STD_LOGIC;
35             clock : in  STD_LOGIC;
36             TrainButton : in  STD_LOGIC; -- Train Button Input --
37             CarButton : in  STD_LOGIC; -- Car Button Input --
38             PedButton : in  STD_LOGIC; -- Pedestrain Button Input --
39             Found : out  STD_LOGIC -- Output To Reset Timer --
40             );
41  end TimerReset;
42
43  architecture Behavioral of TimerReset is
44
45  type StateType is (Train, Car, Pedestrian);
46
47  signal state, nextState : StateType;
48
49  begin
50
51        SynchronousProcess:
52        process (reset, clock)
53        begin
54           if (reset = '1') then -- If Reset Button Is Pressed --
55              state <= Car; -- Set Initial State As Car --
56
57           elsif rising_edge(clock) then -- If Rising Clock Edge --
```

Page 1

TimerReset.vhd Tue Jun 01 12:52:29 2021

```
50                state <= nextState; -- Change Current State To Next State --
59
60             end if;
61          end process SynchronousProcess;
62
63          TimerProcess:
64             process (state, TrainButton, CarButton, PedButton)
65             begin
66
67             Found <= '0'; -- Set Initial Found Output As 0 --
68             nextState <= Car; -- Set Next State As Car --
69
70             case state is
71                when Train => -- When State Is Train --
72
73                   if (CarButton = '1') then -- If CarButton Is Pressed --
74                      Found <= '1'; -- Found Output Is 1 --
75                      nextState <= Car; -- Change Next State To Car --
76
77                   elsif (PedButton = '1') then -- If PedButton Is Pressed --
78                      Found <= '1'; -- Found Output Is 1 --
79                      nextState <= Pedestrian; -- Change Next State To Pedestrian --
80
81                   else -- If None Of The Above Condition Is Satisfied --
82                      nextState <= Train; -- Hold Current State --
83
84                   end if;
85
86                when Car => -- When State Is Car --
87
88                   if (TrainButton = '1') then -- If TrainButton Is Pressed --
89                      Found <= '1'; -- Found Output Is 1 --
90                      nextState <= Train; -- Change Next State To Train --
91
92                   elsif (PedButton = '1') then -- If PedButton Is Pressed --
93                      Found <= '1'; -- Found Output Is 1 --
94                      nextState <= Pedestrian; -- Change Next State To Pedestrian --
95
96                   else -- If None Of The Above Condition Is Satisfied --
97                      nextState <= Car; -- Hold Current State --
98
99                   end if;
100
101               when Pedestrian => -- When State Is Pedestrian --
102
103                  if (TrainButton = '1') then -- If TrainButton Is Pressed --
104                     Found <= '1'; -- Found Output Is 1 --
105                     nextState <= Train; -- Change Next State To Train --
106
107                  elsif (CarButton = '1') then -- If CarButton Is Pressed --
108                     nextState <= Car; -- Change Next State To Car --
109
110                  else -- If None Of The Above Condition Is Satisfied --
111                     nextState <= Pedestrian; -- Hold Current State --
112
113                  end if;
114            end case;
```

Page 2

TimerReset.vhd Tue Jun 01 12:52:29 2021

```
115
116         end process TimerProcess;
117
118   end Behavioral;
119
120
```

Page 3

Figure 29: TimerReset Module Code

SyncButton.vhd **Sat May 22 00:27:07 2021**

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date:    15:57:45 05/20/2021
6   -- Design Name:
7   -- Module Name:    SyncButton - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity SyncButton is
33      Port ( reset : in  STD_LOGIC;
34             clock : in  STD_LOGIC;
35             TrainButton : in  STD_LOGIC; -- Train Button Input --
36             CarButton : in  STD_LOGIC; -- Car Button Input --
37             PedButton : in  STD_LOGIC; -- Pedestrain Button Input --
38             Counter : in  STD_LOGIC_VECTOR (4 downto 0); -- Timer --
39             MotorEnabler : out  STD_LOGIC; -- Enable And Disable Stepper Motor --
40             TrainAction : out  STD_LOGIC; -- Tran Button Output --
41             CarWaitOutput : out  STD_LOGIC; -- Car Output When Train Is And Is Not
   Passing Through --
42             PedWaitOutput : out  STD_LOGIC -- Pedestrain Output When Train Is And Is
   Not Passing Through --
43             );
44  end SyncButton;
45
46  architecture Behavioral of SyncButton is
47
48  type StateType is (GateClose, GateOpenCar, GateOpenPed, TrainPassed, TrainPassing, Car
   , PedestrianPassing);
49
50  signal State, NextState : StateType;
51
52  begin
53
54     SyncButtonProcess:
```

Page 1

SyncButton.vhd **Sat May 22 00:27:07 2021**

```
55          process (reset, clock)
56
57          begin
58             if (reset = '1') then -- If reset is pressed --
59                State <= Car; -- Change The State To Traffic_Green --
60
61             elsif (rising_edge(clock)) then -- If Rising Clock Edge --
62                State <= NextState; -- Change The State To The Next State --
63
64             end if;
65
66          end process;
67
68
69       InputProcess:
70          process (State, Counter, TrainButton, PedButton, CarButton)
71
72          begin
73
74             TrainAction <= '0'; -- Set Initial TrainAction Output To Be 0 --
75             CarWaitOutput <= '0'; -- Set Initial CarWaitOutput Output To Be 0 --
76             PedWaitOutput <= '0'; -- Set Initial PedWaitOutput Output To Be 0 --
77             MotorEnabler <= '0'; -- Set Initial MotorSignalDetector Output To Be 0 --
78
79             NextState <= Car; -- Set Current State To Car --
80
81             case state is
82
83                when GateClose  => -- When The State Is GateClose --
84                   TrainAction <= '1'; -- Set TrainAction Output To Be 1 --
85                   CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
86                   PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
87                   MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --
88
89                   if (Counter = "00101") then -- If Counter Reached "00101" --
90                      NextState <= TrainPassing; -- Change State To TrainPassing --
91
92                   else
93                      NextState <= GateClose; -- If None Of The Above Occur, Hold State --
94
95                   end if;
96
97                when GateOpenCar  => -- When The State Is GateOpenCar --
98                   TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
99                   CarWaitOutput <= '1'; -- Set CarWaitOutput Output To Be 1 --
100                  PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
101                  MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --
102
103                  if (Counter = "00101") then -- If Counter Reached "00101" --
104                     NextState <= Car; -- Change State To Car --
105
106                  else
107                     NextState <= GateOpenCar; -- If None Of The Above Occur, Hold State
    --
108
109                  end if;
110
```

Page 2

SyncButton.vhd Sat May 22 00:27:07 2021

```
111                when GateOpenPed  => -- When The State Is GateOpenPed --
112                   TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
113                   CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
114                   PedWaitOutput <= '1'; -- Set PedWaitOutput Output To Be 1 --
115                   MotorEnabler <= '1'; -- Set MotorEnabler Output To Be 1 --
116
117                   if (Counter = "00101") then -- If Counter Reached "00101" --
118                      NextState <= PedestrianPassing; -- Change State To Car --
119
120                   else
121                      NextState <= GateOpenPed; -- If None Of The Above Occur, Hold State
       --
122
123                   end if;
124
125                when TrainPassed  => -- When The State Is TrainPassed --
126                   TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
127                   CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
128                   PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
129                   MotorEnabler <= '0'; -- Set MotorEnabler Output To Be 0 --
130
131                   if (TrainButton = '1') then -- If TrainButton Is Pressed --
132                      NextState <= TrainPassing; -- Change State To TrainPassing --
133
134                   elsif (CarButton = '1') then -- If CarButton Is Pressed --
135                      NextState <= GateOpenCar; -- Change State To GateOpenCar --
136
137                   elsif (PedButton = '1') then -- If PedButton Is Pressed --
138                      NextState <= GateOpenPed; -- Change State To GateOpenPed --
139
140                   else
141                      NextState <= TrainPassed; -- If None Of The Above Occur, Hold State
       --
142                   end if;
143
144                when TrainPassing  => -- When The State Is TrainPassing --
145                   TrainAction <= '1'; -- Set TrainAction Output To Be 1 --
146                   CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
147                   PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
148                   MotorEnabler <= '0'; -- Set MotorEnabler Output To Be 0 --
149
150                   if (Counter = "11110") then -- If Counter Reached "11110" --
151                      NextState <= TrainPassed; -- Change State To TrainPassed --
152
153                   else
154                      NextState <= TrainPassing; -- If None Of The Above Occur, Hold
       State --
155                   end if;
156
157                when Car => -- When The State Is Car --
158                   TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
159                   CarWaitOutput <= '1'; -- Set CarWaitOutput Output To Be 1 --
160                   PedWaitOutput <= '0'; -- Set PedWaitOutput Output To Be 0 --
161                   MotorEnabler <= '0'; -- Set MotorEnabler Output To Be 0 --
162
163                   if (TrainButton = '1') then -- If TrainButton Is Pressed --
164                      NextState <= GateClose; -- Change State To GateClose --
```

Page 3

SyncButton.vhd Sat May 22 00:27:07 2021

```
165
166                  elsif (PedButton = '1') then -- If PedButton Is Pressed
167                     NextState <= PedestrianPassing; -- Change State To
      PedestrianPassing --
168
169                  else
170                     NextState <= Car; -- If None Of The Above Occur, Hold State --
171                  end if;
172
173               when PedestrianPassing => -- When The State Is PedestrianPassing --
174                  TrainAction <= '0'; -- Set TrainAction Output To Be 0 --
175                  CarWaitOutput <= '0'; -- Set CarWaitOutput Output To Be 0 --
176                  PedWaitOutput <= '1'; -- Set PedWaitOutput Output To Be 1 --
177                  MotorEnabler <= '0';  -- Set MotorEnabler Output To Be 0 --
178
179                  if (Counter = "10100") then -- If Counter Reached "10100" --
180                     NextState <= Car; -- Change State To Car --
181
182                  elsif (TrainButton = '1') then -- If TrainButton Is Pressed --
183                        NextState <= GateClose; -- Change State To GateClose --
184
185                  else
186                     NextState <= PedestrianPassing; -- If None Of The Above Occur, Hold
      State --
187                  end if;
188
189               end case;
190
191      end process;
192   end Behavioral;
```

Page 4

Figure 30: SyncButton Module Code

ISE Synthesis Report

```
Release 14.7 - xst P.20131013 (nt64)
Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
--> Parameter TMPDIR set to xst/projnav.tmp


Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.10 secs

--> Parameter xsthdpdir set to xst


Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.10 secs

--> Reading design: Traffic.prj

TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) Design Hierarchy Analysis
  4) HDL Analysis
  5) HDL Synthesis
     5.1) HDL Synthesis Report
  6) Advanced HDL Synthesis
     6.1) Advanced HDL Synthesis Report
  7) Low Level Synthesis
  8) Partition Report
  9) Final Report

=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================
---- Source Parameters
Input File Name                    : "Traffic.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "Traffic"
Output Format                      : NGC
Target Device                      : CoolRunner2 CPLDs

---- Source Options
Top Module Name                    : Traffic
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
Mux Extraction                     : Yes
Resource Sharing                   : YES

---- Target Options
Add IO Buffers                     : YES
MACRO Preserve                     : YES
XOR Preserve                       : YES
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Keep Hierarchy                     : Yes
Netlist Hierarchy                  : As_Optimized
```

```
RTL Output                          : Yes
Hierarchy Separator                 : /
Bus Delimiter                       : <>
Case Specifier                      : Maintain
Verilog 2001                        : YES

---- Other Options
Clock Enable                        : YES
wysiwyg                             : NO

=========================================================================


=========================================================================
*                           HDL Compilation                             *
=========================================================================
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/SyncButton.vhd" in Library work.
Architecture behavioral of Entity syncbutton is up to date.
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/TimerReset.vhd" in Library work.
Architecture behavioral of Entity timerreset is up to date.
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/Timer.vhd" in Library work.
Architecture behavioral of Entity timer is up to date.
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/MotorController.vhd" in Library
work.
Architecture behavioral of Entity motorcontroller is up to date.
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/TrafficLights.vhd" in Library work.
Architecture behavioral of Entity trafficlights is up to date.
Compiling vhdl file "C:/Xilinx/14.7/TrafficLight_Project/TrafficTopLevel.vhd" in Library
work.
Entity <traffic> compiled.
Entity <traffic> (Architecture <behavioral>) compiled.

=========================================================================
*                     Design Hierarchy Analysis                         *
=========================================================================
Analyzing hierarchy for entity <Traffic> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <SyncButton> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <TimerReset> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <Timer> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <MotorController> in library <work> (architecture
<behavioral>).

Analyzing hierarchy for entity <TrafficLights> in library <work> (architecture
<behavioral>).


=========================================================================
*                            HDL Analysis                               *
=========================================================================
Analyzing Entity <Traffic> in library <work> (Architecture <behavioral>).
Entity <Traffic> analyzed. Unit <Traffic> generated.

Analyzing Entity <SyncButton> in library <work> (Architecture <behavioral>).
Entity <SyncButton> analyzed. Unit <SyncButton> generated.

Analyzing Entity <TimerReset> in library <work> (Architecture <behavioral>).
Entity <TimerReset> analyzed. Unit <TimerReset> generated.
```

```
Analyzing Entity <Timer> in library <work> (Architecture <behavioral>).
Entity <Timer> analyzed. Unit <Timer> generated.

Analyzing Entity <MotorController> in library <work> (Architecture <behavioral>).
Entity <MotorController> analyzed. Unit <MotorController> generated.

Analyzing Entity <TrafficLights> in library <work> (Architecture <behavioral>).
Entity <TrafficLights> analyzed. Unit <TrafficLights> generated.


=========================================================================
*                            HDL Synthesis                              *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <SyncButton>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/SyncButton.vhd".
    Found finite state machine <FSM_0> for signal <State>.
    -----------------------------------------------------------------------
    | States             | 7                                              |
    | Transitions        | 18                                             |
    | Inputs             | 6                                              |
    | Outputs            | 4                                              |
    | Clock              | clock                     (rising_edge)        |
    | Reset              | reset                     (positive)           |
    | Reset type         | asynchronous                                   |
    | Reset State        | car                                            |
    | Power Up State     | gateclose                                      |
    | Encoding           | automatic                                      |
    | Implementation     | automatic                                      |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 Finite State Machine(s).
Unit <SyncButton> synthesized.


Synthesizing Unit <TimerReset>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/TimerReset.vhd".
    Found finite state machine <FSM_1> for signal <state>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 9                                              |
    | Inputs             | 3                                              |
    | Outputs            | 3                                              |
    | Clock              | clock                     (rising_edge)        |
    | Reset              | reset                     (positive)           |
    | Reset type         | asynchronous                                   |
    | Reset State        | car                                            |
    | Power Up State     | train                                          |
    | Encoding           | automatic                                      |
    | Implementation     | automatic                                      |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 Finite State Machine(s).
Unit <TimerReset> synthesized.


Synthesizing Unit <Timer>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/Timer.vhd".
```

```
    Found 1-bit register for signal <MotorPulseOutput>.
    Found 10-bit up counter for signal <counter>.
    Found 10-bit up accumulator for signal <MotorCount>.
    Found 10-bit comparator equal for signal <MotorCount$cmp_eq0000> created at line 84.
    Found 5-bit up counter for signal <timekeeper>.
    Summary:
        inferred   2 Counter(s).
        inferred   1 Accumulator(s).
        inferred   1 D-type flip-flop(s).
        inferred   1 Comparator(s).
Unit <Timer> synthesized.


Synthesizing Unit <MotorController>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/MotorController.vhd".
    Found finite state machine <FSM_2> for signal <State>.
    -----------------------------------------------------------------------
    | States             | 4                                              |
    | Transitions        | 16                                             |
    | Inputs             | 3                                              |
    | Outputs            | 4                                              |
    | Clock              | MotorPulseInput           (rising_edge)        |
    | Reset              | reset                     (positive)           |
    | Reset type         | asynchronous                                   |
    | Reset State        | s0                                             |
    | Power Up State     | s0                                             |
    | Encoding           | automatic                                      |
    | Implementation     | automatic                                      |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 Finite State Machine(s).
Unit <MotorController> synthesized.


Synthesizing Unit <TrafficLights>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/TrafficLights.vhd".
    Found finite state machine <FSM_3> for signal <State>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 9                                              |
    | Inputs             | 3                                              |
    | Outputs            | 3                                              |
    | Clock              | clock                     (rising_edge)        |
    | Reset              | reset                     (positive)           |
    | Reset type         | asynchronous                                   |
    | Reset State        | traffic_green                                  |
    | Power Up State     | flash_amber                                    |
    | Encoding           | automatic                                      |
    | Implementation     | automatic                                      |
    -----------------------------------------------------------------------
    Summary:
        inferred   1 Finite State Machine(s).
Unit <TrafficLights> synthesized.


Synthesizing Unit <Traffic>.
    Related source file is "C:/Xilinx/14.7/TrafficLight_Project/TrafficTopLevel.vhd".
Unit <Traffic> synthesized.


=========================================================================
```

```
HDL Synthesis Report

Macro Statistics
# Counters                                             : 2
 10-bit up counter                                     : 1
 5-bit up counter                                      : 1
# Accumulators                                         : 1
 10-bit up accumulator                                 : 1
# Registers                                            : 1
 1-bit register                                        : 1
# Comparators                                          : 1
 10-bit comparator equal                               : 1

=========================================================================

=========================================================================
*                           Advanced HDL Synthesis                      *
=========================================================================

Analyzing FSM <FSM_3> for best encoding.
Optimizing FSM <TrafficLight/State/FSM> on signal <State[1:2]> with johnson encoding.
---------------------------
 State          | Encoding
---------------------------
 flash_amber    | 00
 traffic_green  | 01
 ped_green      | 11
---------------------------
Analyzing FSM <FSM_2> for best encoding.
Optimizing FSM <StepperModule/State/FSM> on signal <State[1:2]> with sequential encoding.
-------------------
 State | Encoding
-------------------
 s0    | 00
 s1    | 01
 s2    | 11
 s3    | 10
-------------------
Analyzing FSM <FSM_1> for best encoding.
Optimizing FSM <TimerResetting/state/FSM> on signal <state[1:2]> with johnson encoding.
-----------------------
 State      | Encoding
-----------------------
 train      | 00
 car        | 01
 pedestrian | 11
-----------------------
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <ButtonInput/State/FSM> on signal <State[1:3]> with user encoding.
-------------------------------
 State             | Encoding
-------------------------------
 gateclose         | 000
 gateopencar       | 001
 gateopenped       | 010
 trainpassed       | 011
 trainpassing      | 100
 car               | 101
 pedestrianpassing | 110
-------------------------------
```

```
=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# FSMs                                                 : 4
# Counters                                             : 2
 10-bit up counter                                     : 1
 5-bit up counter                                      : 1
# Accumulators                                         : 1
 10-bit up accumulator                                 : 1
# Registers                                            : 1
 Flip-Flops                                            : 1
# Comparators                                          : 1
 10-bit comparator equal                               : 1

=========================================================================

=========================================================================
*                           Low Level Synthesis                         *
=========================================================================

Optimizing unit <Traffic> ...

Optimizing unit <SyncButton> ...
  implementation constraint: INIT=r      : State_FSM_FFd1
  implementation constraint: INIT=r      : State_FSM_FFd2
  implementation constraint: INIT=r      : State_FSM_FFd3

Optimizing unit <TimerReset> ...
  implementation constraint: INIT=r      : state_FSM_FFd1
  implementation constraint: INIT=r      : state_FSM_FFd2

Optimizing unit <MotorController> ...
  implementation constraint: INIT=r      : State_FSM_FFd1
  implementation constraint: INIT=r      : State_FSM_FFd2

Optimizing unit <TrafficLights> ...
  implementation constraint: INIT=r      : State_FSM_FFd1
  implementation constraint: INIT=r      : State_FSM_FFd2

Optimizing unit <Timer> ...

=========================================================================
*                            Partition Report                           *
=========================================================================

Partition Implementation Status
-------------------------------

  No Partitions were found in this design.

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

=========================================================================
*                             Final Report                              *
=========================================================================
Final Results
RTL Top Level Output File Name     : Traffic.ngr
Top Level Output File Name         : Traffic
Output Format                      : NGC
Optimization Goal                  : Speed
```

```
Keep Hierarchy                     : Yes
Target Technology                  : CoolRunner2 CPLDs
Macro Preserve                     : YES
XOR Preserve                       : YES
Clock Enable                       : YES
wysiwyg                            : NO

Design Statistics
# IOs                              : 13

Cell Usage :
# BELS                             : 260
#      AND2                        : 82
#      AND3                        : 9
#      AND4                        : 2
#      AND8                        : 1
#      GND                         : 1
#      INV                         : 91
#      OR2                         : 39
#      OR3                         : 2
#      XOR2                        : 33
# FlipFlops/Latches                : 35
#      FDC                         : 16
#      FDCE                        : 5
#      FDCPE                       : 10
#      FDP                         : 4
# IO Buffers                       : 13
#      IBUF                        : 5
#      OBUF                        : 8
=========================================================================


Total REAL time to Xst completion: 3.00 secs
Total CPU time to Xst completion: 3.07 secs

-->

Total memory usage is 4514324 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Figure 31: ISE Sysnthesis Report

Summary

Design Name	Traffic
Fitting Status	Successful
Software Version	P.20131013
Device Used	XC2C64A-7-VQ44
Date	5-24-2021, 5:35PM

RESOURCES SUMMARY

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
48/64 (75%)	129/224 (58%)	35/64 (55%)	13/33 (40%)	76/160 (48%)

PIN RESOURCES

Signal Type	Required	Mapped	Pin Type	Used	Total
Input	3	3	I/O	9	26
Output	8	8	GCK/IO	1	3
Bidirectional	0	0	GTS/IO	2	4
GCK	1	1	GSR/IO	1	1
GTS	0	0	DGE/IO	0	-1
GSR	1	1			

GLOBAL RESOURCES

Signal mapped onto global clock net (GCK2)	Clock
Signal mapped onto global output enable net (GSR)	Reset

back to top print page

Figure 32: ISE Summary Report

Appendix

TrafficTopLevel.vhd **Fri May 28 13:19:13 2021**

```
1   ----------------------------------------------------------------------------------
2   -- TrafficTopLevel.vhd
3   --
4   -- Traffic light system to control an intersection
5   --
6   --
7   ----------------------------------------------------------------------------------
8   library IEEE;
9   use IEEE.STD_LOGIC_1164.ALL;
10  use IEEE.STD_LOGIC_ARITH.ALL;
11  use IEEE.STD_LOGIC_UNSIGNED.ALL;
12
13  entity Traffic is
14     Port (   Reset : in STD_LOGIC;
15              Clock : in STD_LOGIC;
16
17              -- Car and pedestrian buttons
18              Train : in STD_LOGIC; -- Train on EW road
19              CarNS : in STD_LOGIC; -- Car on NS road
20              PedNS : in STD_LOGIC; -- Pedestrian moving NS (crossing EW road)
21
22              -- Light control
23              HLights : out STD_LOGIC_VECTOR (1 downto 0); -- controls EW lights
24              VLights : out STD_LOGIC_VECTOR (1 downto 0); -- controls NS lights
25              Motor    : out STD_LOGIC_VECTOR (3 downto 0)   -- controls Motor
26           );
27  end entity Traffic;
28
29  architecture Behavioral of Traffic is
30     COMPONENT SyncButton
31        PORT ( reset : in  STD_LOGIC;
32             clock : in  STD_LOGIC;
33             TrainButton : in  STD_LOGIC; -- Train Button Input --
34             CarButton : in  STD_LOGIC; -- Car Button Input --
35             PedButton : in  STD_LOGIC; -- Pedestrain Button Input --
36             Counter : in  STD_LOGIC_VECTOR (4 downto 0); -- Timer --
37             MotorEnabler : out  STD_LOGIC; -- Enable And Disable Stepper Motor --
38             TrainAction : out  STD_LOGIC; -- Tran Button Output --
39             CarWaitOutput : out  STD_LOGIC; -- Car Output When Train Is And Is Not
   Passing Through --
40             PedWaitOutput : out  STD_LOGIC -- Pedestrain Output When Train Is And Is
   Not Passing Through --
41             );
42        END COMPONENT;
43
44     COMPONENT TimerReset
45        PORT ( reset : in  STD_LOGIC;
46             clock : in  STD_LOGIC;
47             TrainButton : in  STD_LOGIC; -- Train Button Input --
48             CarButton : in  STD_LOGIC; -- Car Button Input --
49             PedButton : in  STD_LOGIC; -- Pedestrain Button Input --
50             Found : out  STD_LOGIC -- Output To Reset Timer --
51             );
52        END COMPONENT;
53
54     COMPONENT Timer
55        PORT ( reset : in  STD_LOGIC;
```

Page 1

TrafficTopLevel.vhd **Fri May 28 13:19:13 2021**

```
56              clock : in  STD_LOGIC;
57              timer_reset : in  STD_LOGIC; -- To Reset Timer --
58              FourHzPulse : out  STD_LOGIC; -- Four Hz Pulse --
59              MotorPulseOutput : out STD_LOGIC; -- Pulse For Stepper Motor --
60              CounterOutput : out  STD_LOGIC_VECTOR (4 downto 0) -- Timer Output --
61              );
62         END COMPONENT;
63
64      COMPONENT MotorController
65         PORT ( reset : in  STD_LOGIC;
66              MotorPulseInput : in  STD_LOGIC; -- Pulse For Stepper Motor --
67              Enable : in  STD_LOGIC; -- To Enable The Motor To Turn --
68              Clockwise : in  STD_LOGIC; -- To Choose Direction To Turn --
69              MotorEnablerInput : in  STD_LOGIC; -- To Enable Stepper Motor --
70              MotorOutput : out  STD_LOGIC_VECTOR (3 downto 0) -- Output for Motor --
71              );
72         END COMPONENT;
73
74      COMPONENT TrafficLights
75         PORT ( reset : in  STD_LOGIC;
76              clock : in  STD_LOGIC;
77              FourHzPulse : in  STD_LOGIC; -- 4Hz Pulse --
78              Flash : in  STD_LOGIC; -- Train Button is Pressed --
79              TrafficGreen : in  STD_LOGIC; -- Car Button is Pressed --
80              PedGreen : in  STD_LOGIC; -- Pedestrian Button is Pressed --
81              MotorEnable : out  STD_LOGIC; -- Enable The Stepper Motor To Rotate --
82              MotorClockwise : out  STD_LOGIC; -- Direction Of Rotation For Stepper
    Motor--
83              HTrafficLightOutput : out  STD_LOGIC_VECTOR (1 downto 0); -- Output for
    Horizontal Traffic Light --
84              VTrafficLightOutput : out  STD_LOGIC_VECTOR (1 downto 0) -- Output for
    Vertical Traffic Light --
85              );
86         END COMPONENT;
87
88      signal TrainInputButton : std_logic;
89      signal CarInputButton : std_logic;
90      signal PedInputButton : std_logic;
91      signal ResetTimer : std_logic;
92      signal MotorEnableDetector : std_logic;
93      signal MotorClockwiseDetector : std_logic;
94      signal MotorPulseOutputDetector : std_logic;
95      signal FourHzOutput : std_logic;
96      signal MotorEnablerDetector : std_logic;
97      signal TimerOutput : std_logic_vector (4 downto 0);
98
99      begin
100
101     --Insert your code here --
102
103     ButtonInput : SyncButton PORT MAP (
104        reset => Reset,
105        clock => Clock,
106        TrainButton => Train,
107        CarButton => CarNS,
108        PedButton => PedNS,
109        Counter => TimerOutput,
```

Page 2

TrafficTopLevel.vhd **Fri May 28 13:19:13 2021**

```
110         MotorEnabler => MotorEnablerDetector,
111         TrainAction => TrainInputButton,
112         CarWaitOutput => CarInputButton,
113         PedWaitOutput => PedInputButton
114      );
115
116      TimerResetting : TimerReset PORT MAP (
117         reset => Reset,
118         clock => Clock,
119         TrainButton => TrainInputButton,
120         CarButton => CarInputButton,
121         PedButton => PedInputButton,
122         Found => ResetTimer
123      );
124
125      SystemTimer : Timer PORT MAP (
126         reset => Reset,
127         clock => Clock,
128         timer_reset => ResetTimer,
129         FourHzPulse => FourHzOutput,
130         MotorPulseOutput => MotorPulseOutputDetector,
131         CounterOutput => TimerOutput
132      );
133
134      StepperModule : MotorController PORT MAP (
135         reset => Reset,
136         MotorPulseInput => MotorPulseOutputDetector,
137         Enable => MotorEnableDetector,
138         Clockwise => MotorClockwiseDetector,
139         MotorEnablerInput => MotorEnablerDetector,
140         MotorOutput => Motor
141      );
142
143      TrafficLight : TrafficLights PORT MAP (
144         reset => Reset,
145         clock => Clock,
146         FourHzPulse => FourHzOutput,
147         Flash => TrainInputButton,
148         TrafficGreen => CarInputButton,
149         PedGreen => PedInputButton,
150         MotorEnable => MotorEnableDetector,
151         MotorClockwise => MotorClockwiseDetector,
152         HTrafficLightOutput => HLights,
153         VTrafficLightOutput => VLights
154      );
155
156   end architecture Behavioral;
```

Page 3

Figure 33: TrafficTopLevel.vhd Code

TopLevel.ucf **Fri May 28 13:19:26 2021**

```
1   NET "Clock" LOC = "p1"|IOSTANDARD=LVCMOS33; # CMOD-P35 - GCK2, Global Clock Input
2
3   #System level constraints
4   Net Clock TNM_NET = Clock;
5   Timespec TS_Clock = PERIOD Clock 10us; #adjust to suit Clock, 10us->100kHz
6
7   #Reset pin
8   NET "Reset" LOC = "p30"|IOSTANDARD=LVCMOS33; # CMOD-P18 - GCK2, Global Set/Reset
9   Net "Reset" TIG;
10
11  NET "Train" LOC = "p33"|IOSTANDARD=LVCMOS33; # CMOD-P24 - GTS0, Global Tristate Control
12  NET "CarNS" LOC = "p34"|IOSTANDARD=LVCMOS33; # CMOD-P25 - GTS1, Global Tristate Control
13
14  NET "PedNS" LOC = "p37"|IOSTANDARD=LVCMOS33; # CMOD-P27 - I/O
15
16  NET "HLights<1>" LOC = "p39"|IOSTANDARD=LVCMOS33; # CMOD-P29 - I/O
17  NET "HLights<0>" LOC = "p40"|IOSTANDARD=LVCMOS33; # CMOD-P30 - I/O
18
19  NET "VLights<1>" LOC = "p41"|IOSTANDARD=LVCMOS33; # CMOD-P31 - I/O
20  NET "VLights<0>" LOC = "p42"|IOSTANDARD=LVCMOS33; # CMOD-P32 - I/O
21
22  NET "Motor<0>" LOC = "p21"|IOSTANDARD=LVCMOS33; # CMOD-P9 - I/O
23  NET "Motor<1>" LOC = "p20"|IOSTANDARD=LVCMOS33; # CMOD-P10 - I/O
24  NET "Motor<2>" LOC = "p19"|IOSTANDARD=LVCMOS33; # CMOD-P11 - I/O
25  NET "Motor<3>" LOC = "p18"|IOSTANDARD=LVCMOS33; # CMOD-P12 - I/O
```

Figure 34: TopLevel.ucf Code