```
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;
```

```
 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 --
```

```
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;
```