Final Project: UART Design

CEG 3155

Abilaash Uthayachandran

#300116640

Nutan Nimkar

#300127333

Due Date: 12/08/2021

Objective

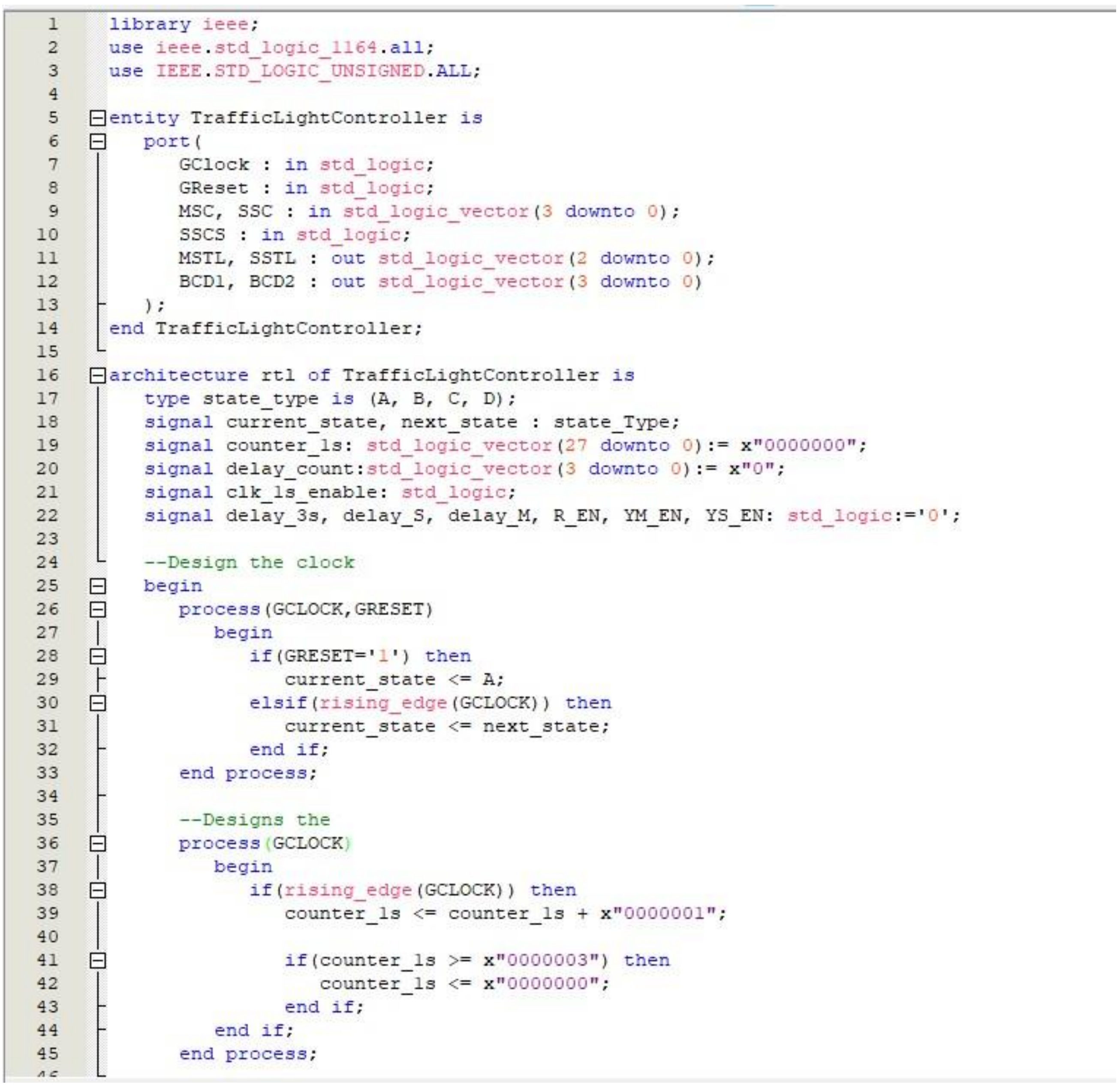
The objective of this project is to design and build a complete UART in VHDL:

- Design, realize and test transmitter and receiver modules.
- Design, realize and test a baud rate generator.
- Demonstrate a complete understanding for the design of a UART and its interface in a real-time system.

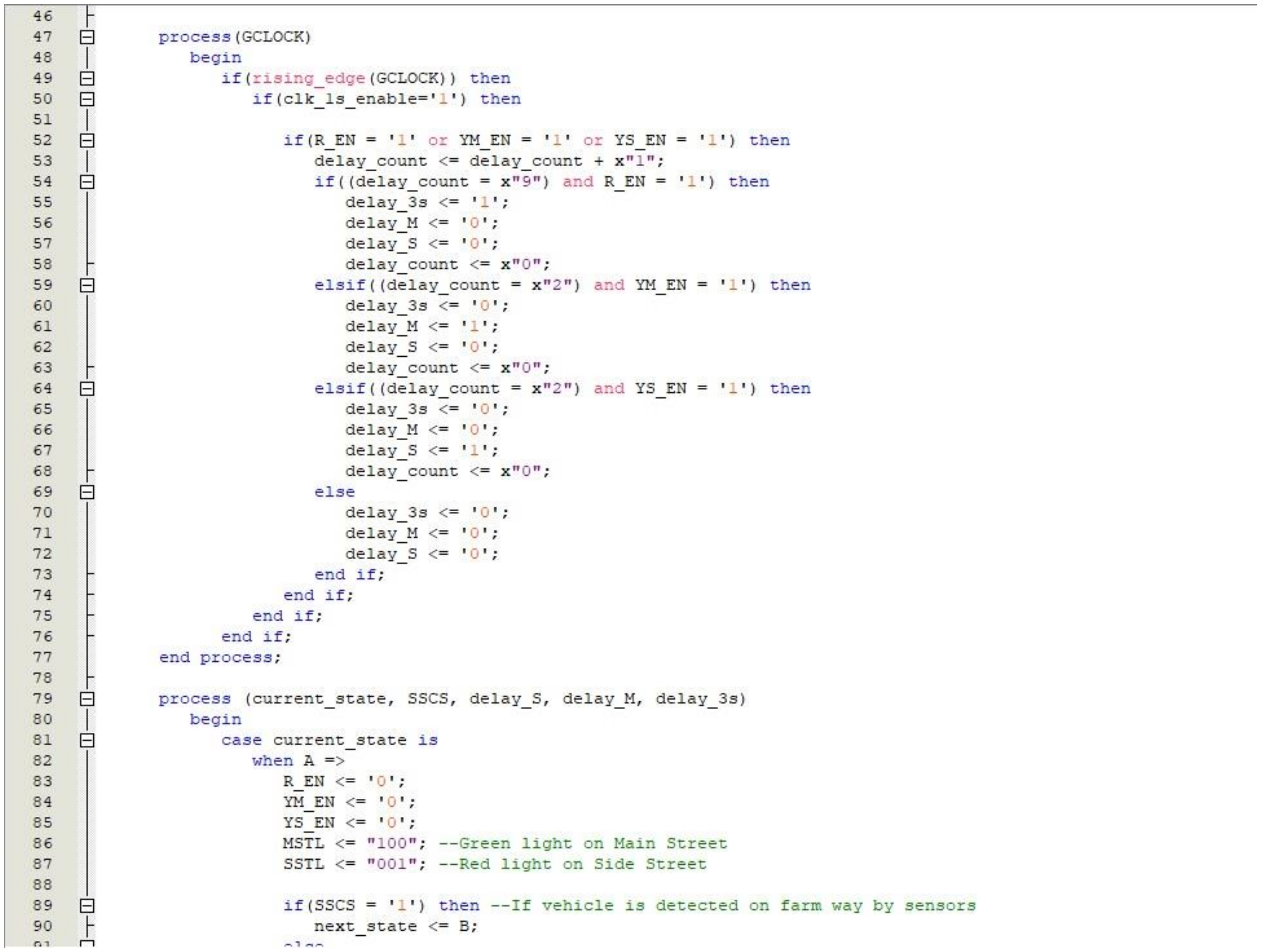
Problem and Solution

The problem at hand is to implement a UART, by realizing the UART components (receiver and transmitter) and by realizing the UART control circuits and finally by connecting the lot to real sensors and actuators. All code was done in VHDL in structural format. We directly jumped onto the coding aspect of the project due to not having enough time to create the FSM. We used the designs provided in the lab manual to start the implementation of the project

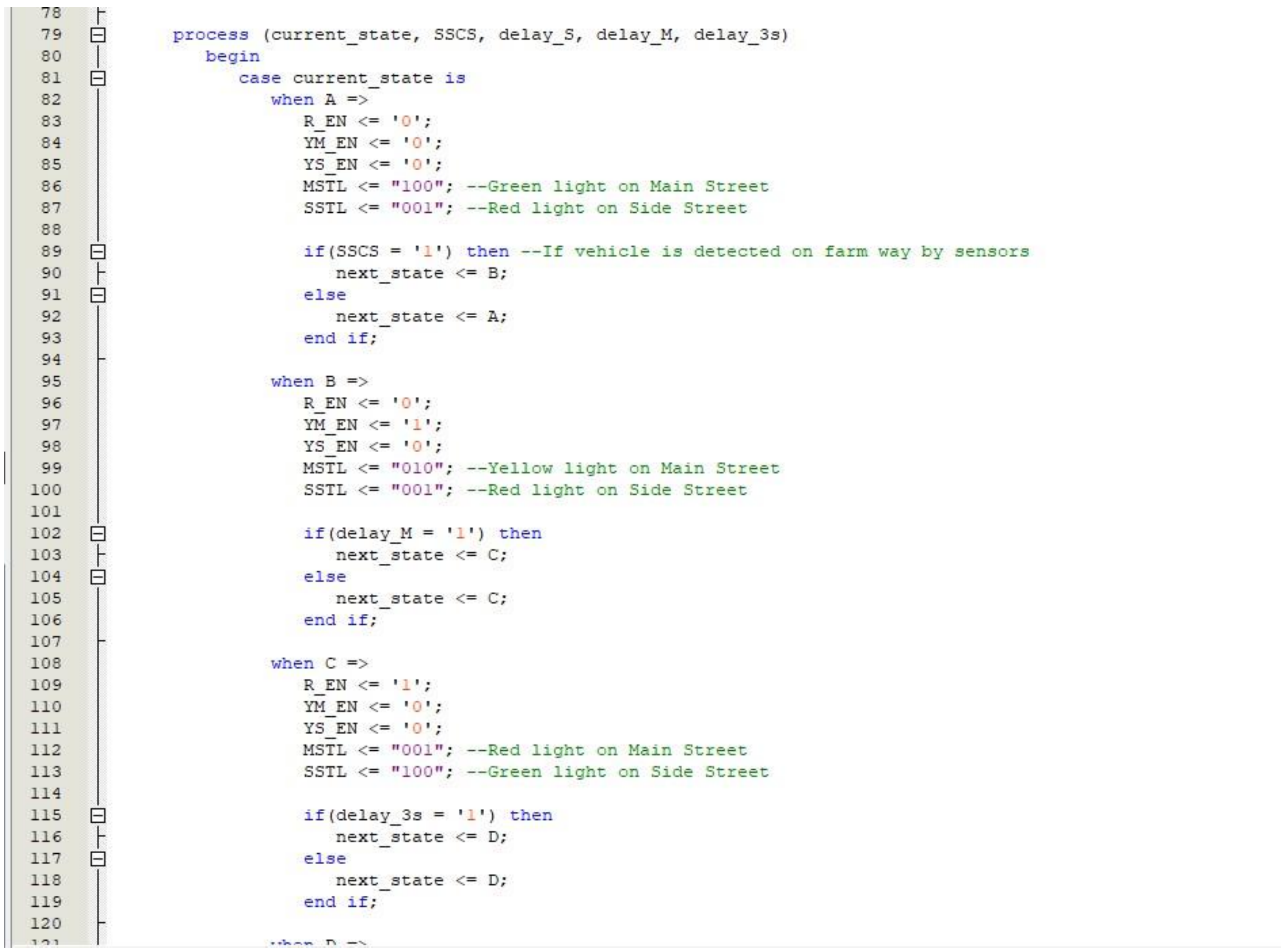
Code



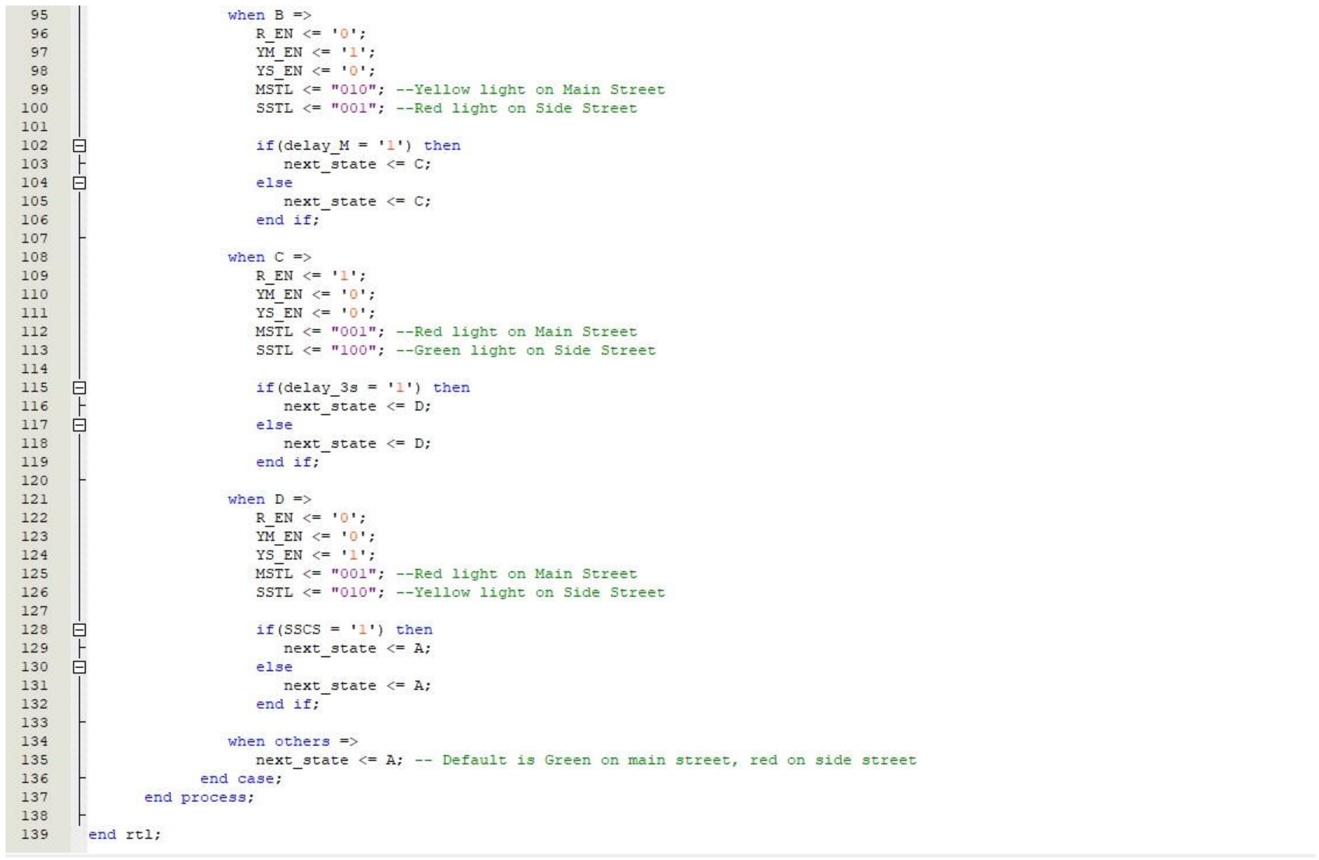
```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity TrafficLightController is
    port(
        GClock : in std_logic;
        GReset : in std_logic;
        MSC, SSC : in std_logic_vector(3 downto 0);
        SSCS : in std_logic;
        MSTL, SSTL : out std_logic_vector(2 downto 0);
        BCD1, BCD2 : out std_logic_vector(3 downto 0)
    );
end TrafficLightController;

architecture rtl of TrafficLightController is
    type state_type is (A, B, C, D);
    signal current_state, next_state : state_Type;
    signal counter_1s: std_logic_vector(27 downto 0):= x"0000000";
    signal delay_count:std_logic_vector(3 downto 0):= x"0";
    signal clk_1s_enable: std_logic;
    signal delay_3s, delay_S, delay_M, R_EN, YM_EN, YS_EN: std_logic:='0';

    --Design the clock
    begin
        process(GCLOCK,GRESET)
            begin
                if(GRESET='1') then
                    current_state <= A;
                elsif(rising_edge(GCLOCK)) then
                    current_state <= next_state;
                end if;
        end process;

        --Designs the
        process(GCLOCK)
            begin
                if(rising_edge(GCLOCK)) then
                    counter_1s <= counter_1s + x"0000001";

                    if(counter_1s >= x"0000003") then
                        counter_1s <= x"0000000";
                    end if;
                end if;
        end process;
```



```vhdl

process(GCLOCK)
   begin
      if(rising_edge(GCLOCK)) then
         if(clk_1s_enable='1') then

            if(R_EN = '1' or YM_EN = '1' or YS_EN = '1') then
               delay_count <= delay_count + x"1";
               if((delay_count = x"9") and R_EN = '1') then
                  delay_3s <= '1';
                  delay_M <= '0';
                  delay_S <= '0';
                  delay_count <= x"0";
               elsif((delay_count = x"2") and YM_EN = '1') then
                  delay_3s <= '0';
                  delay_M <= '1';
                  delay_S <= '0';
                  delay_count <= x"0";
               elsif((delay_count = x"2") and YS_EN = '1') then
                  delay_3s <= '0';
                  delay_M <= '0';
                  delay_S <= '1';
                  delay_count <= x"0";
               else
                  delay_3s <= '0';
                  delay_M <= '0';
                  delay_S <= '0';
               end if;
            end if;
         end if;
      end if;
end process;

process (current_state, SSCS, delay_S, delay_M, delay_3s)
   begin
      case current_state is
         when A =>
            R_EN <= '0';
            YM_EN <= '0';
            YS_EN <= '0';
            MSTL <= "100"; --Green light on Main Street
            SSTL <= "001"; --Red light on Side Street

            if(SSCS = '1') then --If vehicle is detected on farm way by sensors
               next_state <= B;
```



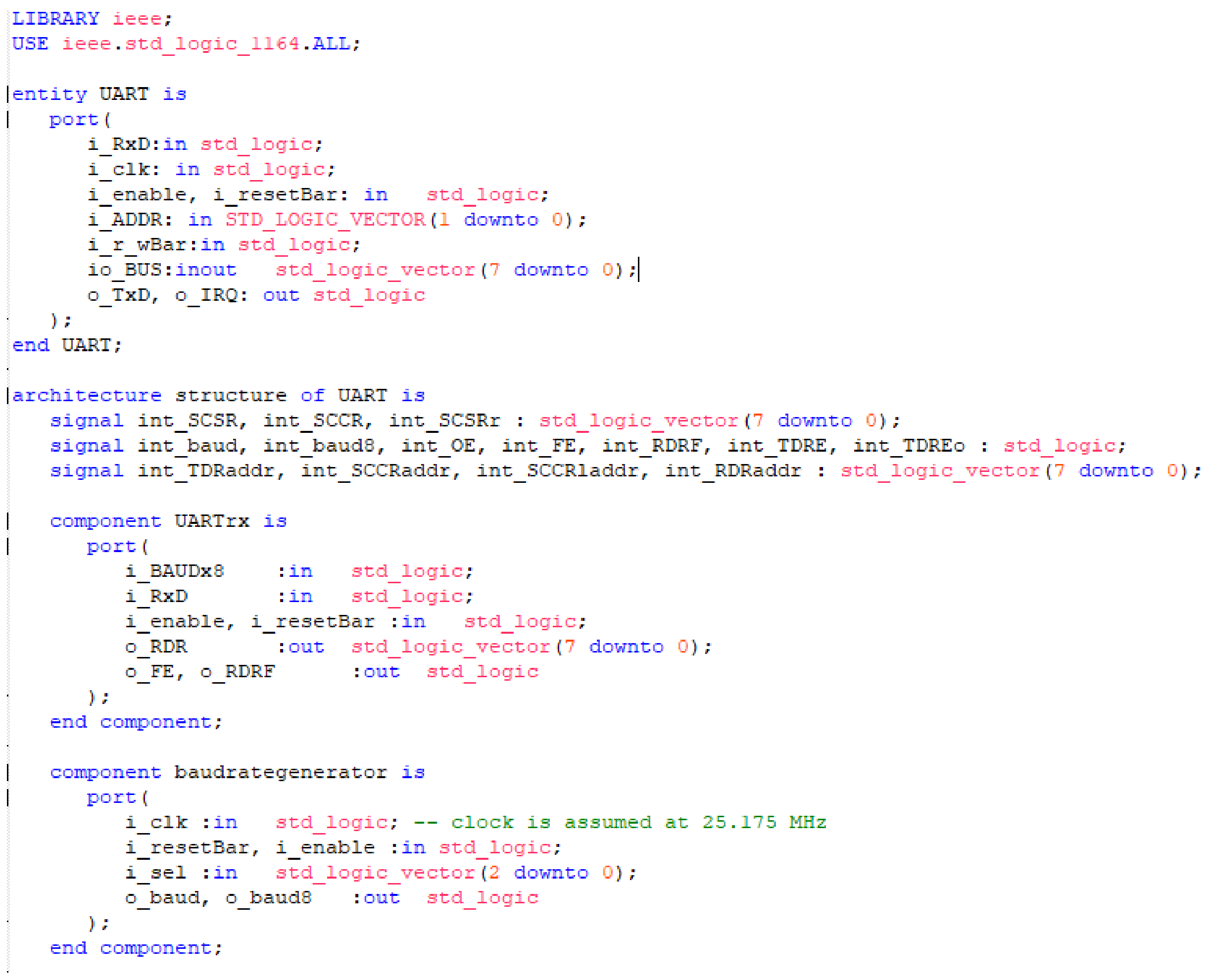
```vhdl

process (current_state, SSCS, delay_S, delay_M, delay_3s)
   begin
      case current_state is
         when A =>
            R_EN <= '0';
            YM_EN <= '0';
            YS_EN <= '0';
            MSTL <= "100"; --Green light on Main Street
            SSTL <= "001"; --Red light on Side Street

            if(SSCS = '1') then --If vehicle is detected on farm way by sensors
               next_state <= B;
            else
               next_state <= A;
            end if;

         when B =>
            R_EN <= '0';
            YM_EN <= '1';
            YS_EN <= '0';
            MSTL <= "010"; --Yellow light on Main Street
            SSTL <= "001"; --Red light on Side Street

            if(delay_M = '1') then
               next_state <= C;
            else
               next_state <= C;
            end if;

         when C =>
            R_EN <= '1';
            YM_EN <= '0';
            YS_EN <= '0';
            MSTL <= "001"; --Red light on Main Street
            SSTL <= "100"; --Green light on Side Street

            if(delay_3s = '1') then
               next_state <= D;
            else
               next_state <= D;
            end if;

```



```
 95                 when B =>
 96                    R_EN <= '0';
 97                    YM_EN <= '1';
 98                    YS_EN <= '0';
 99                    MSTL <= "010"; --Yellow light on Main Street
100                    SSTL <= "001"; --Red light on Side Street
101
102                    if(delay_M = '1') then
103                       next_state <= C;
104                    else
105                       next_state <= C;
106                    end if;
107
108                 when C =>
109                    R_EN <= '1';
110                    YM_EN <= '0';
111                    YS_EN <= '0';
112                    MSTL <= "001"; --Red light on Main Street
113                    SSTL <= "100"; --Green light on Side Street
114
115                    if(delay_3s = '1') then
116                       next_state <= D;
117                    else
118                       next_state <= D;
119                    end if;
120
121                 when D =>
122                    R_EN <= '0';
123                    YM_EN <= '0';
124                    YS_EN <= '1';
125                    MSTL <= "001"; --Red light on Main Street
126                    SSTL <= "010"; --Yellow light on Side Street
127
128                    if(SSCS = '1') then
129                       next_state <= A;
130                    else
131                       next_state <= A;
132                    end if;
133
134                 when others =>
135                    next_state <= A; -- Default is Green on main street, red on side street
136              end case;
137        end process;
138
139  end rtl;
```

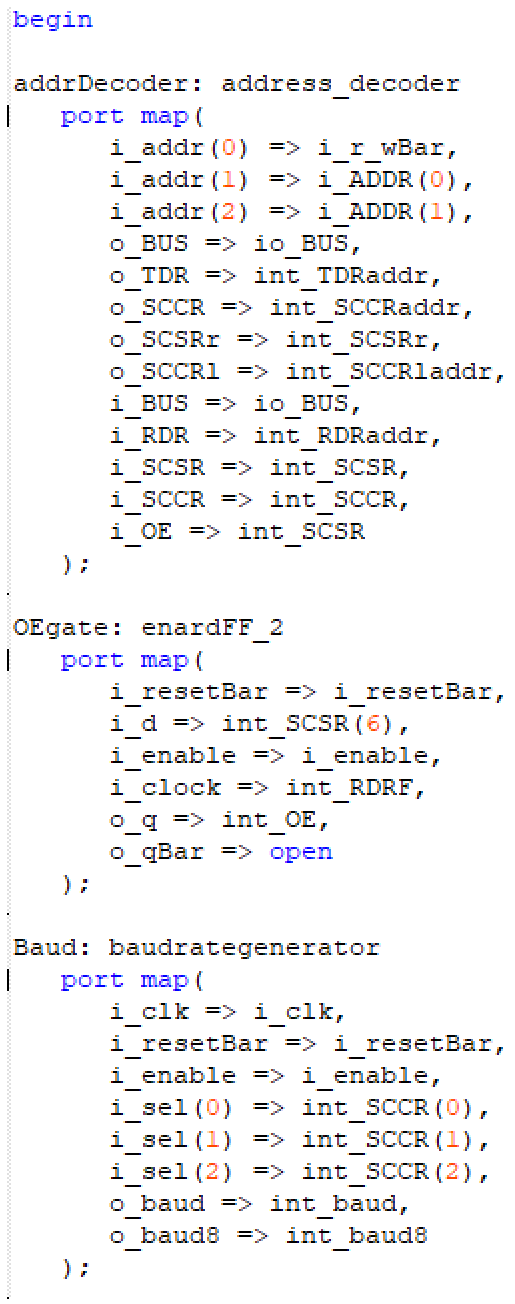
The code shown here contains all the components for the controller. The timer is designed using the global clock input with a 1 second counter variable to count when MSC or SSC is reached. The state transitions are implemented on line 25 where the current and next state are stored, and the transition occurs. The block of code after line 79 is to change all the light inputs to whatever state the code is on. Also, when the state changes, the next state is determined here depending on what the output of the previous state is and what the current state is. The “Enable” signals are used for checking the state of the signal and the MSTL and SSTL are then derived using simple logic from the state table in Figure 1. After each state is run through, the code will output the state of both signals and then use those outputs to determine the state of the next signals when the timer reaches 0.

The code shown below is for the UART which takes all the devices we have designed and connects them altogether to design the UART. The main components implemented in the UART are the following: transmitter, receiver, address decoder, asynchronous register, enabled D flip flop, and the baud rate generator.



```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity UART is
   port(
      i_RxD:in std_logic;
      i_clk: in std_logic;
      i_enable, i_resetBar: in   std_logic;
      i_ADDR: in STD_LOGIC_VECTOR(1 downto 0);
      i_r_wBar:in std_logic;
      io_BUS:inout   std_logic_vector(7 downto 0);
      o_TxD, o_IRQ: out std_logic
   );
end UART;

architecture structure of UART is
   signal int_SCSR, int_SCCR, int_SCSRr : std_logic_vector(7 downto 0);
   signal int_baud, int_baud8, int_OE, int_FE, int_RDRF, int_TDRE, int_TDREo : std_logic;
   signal int_TDRaddr, int_SCCRaddr, int_SCCRladdr, int_RDRaddr : std_logic_vector(7 downto 0);

   component UARTrx is
      port(
         i_BAUDx8     :in   std_logic;
         i_RxD        :in   std_logic;
         i_enable, i_resetBar :in   std_logic;
         o_RDR        :out  std_logic_vector(7 downto 0);
         o_FE, o_RDRF       :out  std_logic
      );
   end component;

   component baudrategenerator is
      port(
         i_clk :in   std_logic; -- clock is assumed at 25.175 MHz
         i_resetBar, i_enable :in std_logic;
         i_sel :in   std_logic_vector(2 downto 0);
         o_baud, o_baud8    :out  std_logic
      );
   end component;
```

```
component UARTtx is
   port(
      i_BAUD          :in   std_logic;
      i_enable, i_resetBar :in   std_logic;
      i_TDR        :in   std_logic_vector(7 downto 0);
      i_loadTDR       :in   std_logic;
      i_TDRE          :in   std_logic;
      o_TxD        :out  std_logic;
      o_TDRE          :out  std_logic
   );
end component;

component asyncReg8b IS
   PORT(
      i_d, i_load :in   std_logic_vector(7 downto 0);
      o_q :out std_logic_vector(7 downto 0)
   );
end component;

component enardFF_2 IS
   PORT(
      i_resetBar  : IN  STD_LOGIC;
      i_d       : IN  STD_LOGIC;
      i_enable : IN  STD_LOGIC;
      i_clock     : IN  STD_LOGIC;
      o_q, o_qBar : OUT STD_LOGIC
   );
end component;

component address_decoder IS
   PORT(
      i_addr: in STD_LOGIC_VECTOR(2 downto 0);
      o_BUS, o_TDR , o_SCCR: out STD_LOGIC_VECTOR(7 downto 0);
      o_SCSRr, o_SCCR1: OUT STD_LOGIC_VECTOR(7 downto 0);
      i_BUS, i_RDR, i_SCSR, i_SCCR, i_OE: in STD_LOGIC_VECTOR(7 downto 0)
   );
end component;
```

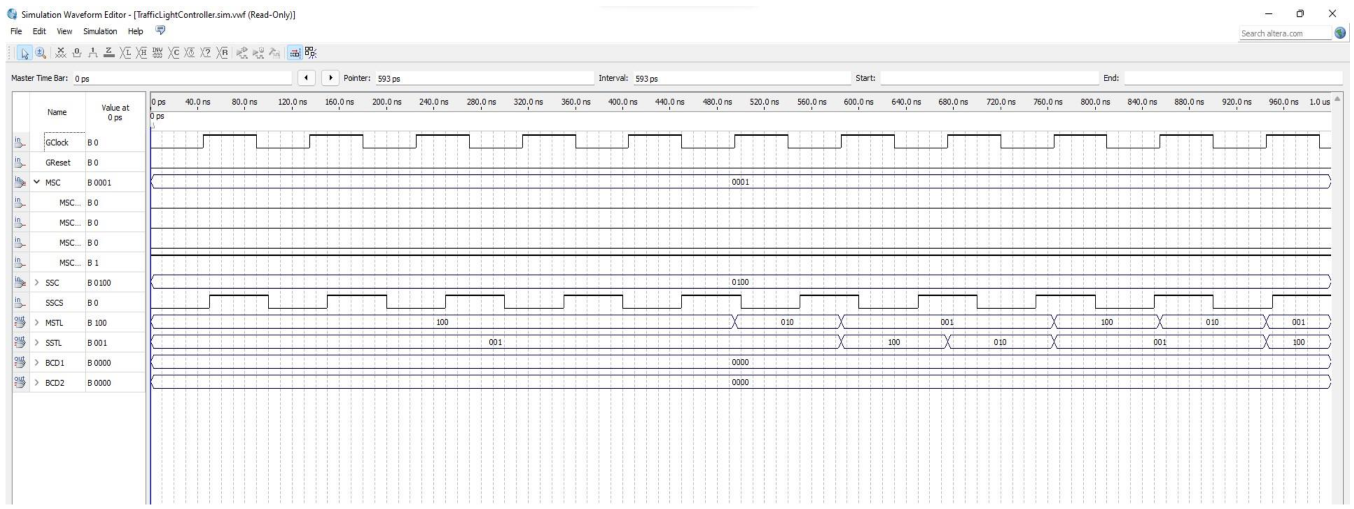


```
begin

addrDecoder: address_decoder
    port map(
        i_addr(0) => i_r_wBar,
        i_addr(1) => i_ADDR(0),
        i_addr(2) => i_ADDR(1),
        o_BUS => io_BUS,
        o_TDR => int_TDRaddr,
        o_SCCR => int_SCCRaddr,
        o_SCSRr => int_SCSRr,
        o_SCCR1 => int_SCCR1addr,
        i_BUS => io_BUS,
        i_RDR => int_RDRaddr,
        i_SCSR => int_SCSR,
        i_SCCR => int_SCCR,
        i_OE => int_SCSR
    );

OEgate: enardFF_2
    port map(
        i_resetBar => i_resetBar,
        i_d => int_SCSR(6),
        i_enable => i_enable,
        i_clock => int_RDRF,
        o_q => int_OE,
        o_qBar => open
    );

Baud: baudrategenerator
    port map(
        i_clk => i_clk,
        i_resetBar => i_resetBar,
        i_enable => i_enable,
        i_sel(0) => int_SCCR(0),
        i_sel(1) => int_SCCR(1),
        i_sel(2) => int_SCCR(2),
        o_baud => int_baud,
        o_baud8 => int_baud8
    );
```

```
Rx: UARTrx
   port map(
      i_BAUDx8 => int_baud8,
      i_RxD => i_RxD,
      i_enable => i_enable,
      i_resetBar => i_resetBar,
      o_RDR => int_RDRaddr,
      o_FE => int_FE,
      o_RDRF => int_RDRF
   );

Tx: UARTtx
   port map(
      i_BAUD => int_baud,
      i_enable => i_enable,
      i_resetBar => i_resetBar,
      i_TDR => int_TDRaddr,
      i_loadTDR => int_TDRE,
      i_TDRE => int_TDRE,
      o_TxD => o_TxD,
      o_TDRE => int_TDREo
   );

SCSR: asyncReg8b
   port map(
      i_d(0) => int_FE and not(int_SCSRr(0)),
      i_d(1) => int_OE and not(int_SCSRr(1)),
      i_d(2) => '0',
      i_d(3) => '0',
      i_d(4) => '0',
      i_d(5) => '0',
      i_d(6) => int_RDRF and not(int_SCSRr(6)),
      i_d(7) => ((int_TDRE or int_TDREo) and not(int_SCSRr(7))) or not(i_resetBar),
      i_load(0) => (int_FE or not(i_resetBar)) or int_SCSRr(0),
      i_load(1) => (int_OE or not(i_resetBar)) or int_SCSRr(1),
      i_load(2) => ('0' or not(i_resetBar)),
      i_load(3) => ('0' or not(i_resetBar)),
      i_load(4) => ('0' or not(i_resetBar)),
      i_load(5) => ('0' or not(i_resetBar)),
      i_load(6) => (int_RDRF or not(i_resetBar)) or int_SCSRr(6),
      i_load(7) => (int_TDREo or not(i_resetBar)) or int_SCSRr(7),
      o_q => int_SCSR
   );
```

```
    int_TDRE <= int_SCSR(7);

SCCR: asyncReg8b
    port map(
        i_d(0) => int_SCCRaddr(0) and i_resetBar,
        i_d(1) => int_SCCRaddr(1) and i_resetBar,
        i_d(2) => int_SCCRaddr(2) and i_resetBar,
        i_d(3) => '0',
        i_d(4) => '0',
        i_d(5) => '0',
        i_d(6) => int_SCCRaddr(6) and i_resetBar,
        i_d(7) => int_SCCRaddr(7) and i_resetBar,
        i_load => int_SCCRladdr,
        o_q => int_SCCR
    );

    o_IRQ <= (int_SCCR(7) and int_SCSR(7))or(int_SCCR(6) and(int_SCSR(1) or int_SCSR(6)));

end structure;
```

Verification

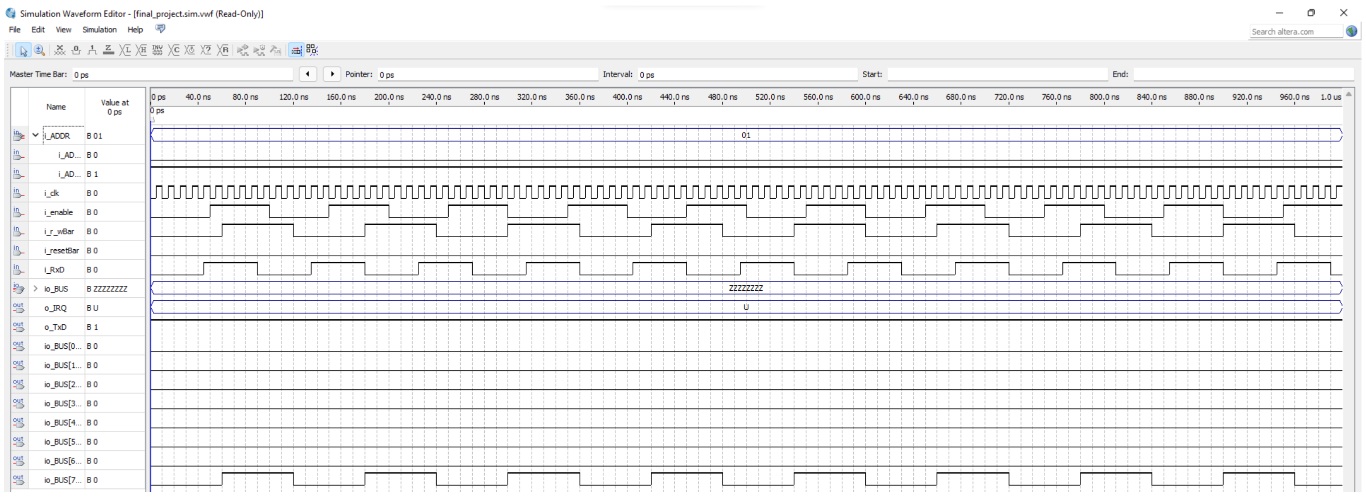

Simulation Waveform Editor - [TrafficLightController.sim.vwf (Read-Only)]

Master Time Bar: 0 ps   Pointer: 593 ps   Interval: 593 ps   Start: End:

Name	Value at 0 ps
GClock	B 0
GReset	B 0
MSC	B 0001
MSC...	B 0
MSC...	B 0
MSC...	B 0
MSC...	B 1
SSC	B 0100
SSCS	B 0
MSTL	B 100
SSTL	B 001
BCD1	B 0000
BCD2	B 0000

File Edit View Simulation Help
0 ps 40.0 ns 80.0 ns 120.0 ns 160.0 ns 200.0 ns 240.0 ns 280.0 ns 320.0 ns 360.0 ns 400.0 ns 440.0 ns 480.0 ns 520.0 ns 560.0 ns 600.0 ns 640.0 ns 680.0 ns 720.0 ns 760.0 ns 800.0 ns 840.0 ns 880.0 ns 920.0 ns 960.0 ns 1.0 us
0001
0100
100 010 001 100 010 001
001 100 010 001 100
0000
0000

This is the waveform simulation of the traffic light counter when MSC is given an input of “0001” and SSC is given an input of “0100”.



Simulation of the UART using random variable inputs to display what the output would look like.

Conclusion

In this lab we implemented our UART using the devices such as the transmitter, receiver, address decoder, etc. however, due to some errors and time constraints, we were unable to complete the UART FSM. The project has a working UART with no errors, and the lab 3 traffic light controller still works with a working result. Our FSM design was not completed as we did not know how the ports should be linked between the devices. From our simulations, we can see that the traffic light returns our desired output with the clock being synced with the UART. What we learned from the project is how to design the transmitter/receiver in VHDL with an asynchronous clock. The aspect of the UART is the solid foundation of the project whereas the FSM is the microcontroller which connects the UART and the traffic light to return and print debug messages on every state change.