Ankara University Faculty of Engineering Computer Engineering Department 2022-23 Fall Semester Microprocessors & Embedded Systems Term Project

Course Code: COM3525-B

Problem:

To implement traffic lights in VHDL.

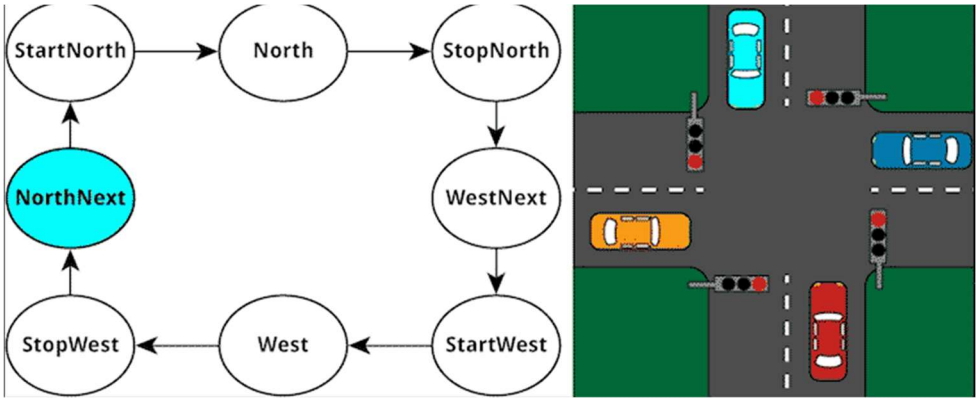
Ülke Eren Aktaş 21290742

Mehmet Bayram Alpay 20290310

The purpose and problem

The aim of the project is to design the traffic lights we encounter in our daily life in a computer environment using VHDL. It is necessary to design a system with traffic lights so that vehicles do not have an accident in places where there are road junctions. In systems with more than one traffic light, one traffic light must be green and the other traffic light red, as in our project. Also, before the green light comes on, the yellow light should come on as a precaution. Apart from this, more than one color should not be lit at the same time in a traffic light. For such synchronization settings, we need to design a model and determine the duration of the lights and the system must run in the same flow all the time.

What We Learned




```
architecture rtl of T23_TrafficLights is

    -- Enumerated type declaration and state signal declaration
    type t_State is (NorthNext, StartNorth, North, StopNorth,
                     WestNext, StartWest, West, StopWest);
    signal State : t_State;

    -- Counter for counting clock periods, 1 minute max
    signal Counter : integer range 0 to ClockFrequencyHz * 60;
```

Its better to directly nosedive to the code since it's not too complicated and long. First we declare the states which our traffic lights are going to behave upon and the counter. The ClockFrequencyHz value is accesseded through our testbench file which is the main code. We declare and reference its value from that file.

```
architecture sim of piptb is

    -- We are using a low clock frequency to speed up the simulation
    constant ClockFrequencyHz : integer := 100; -- 100 Hz
    constant ClockPeriod : time := 1000 ms / ClockFrequencyHz;

    signal Clk          : std_logic := '1';
    signal nRst         : std_logic := '0';
    signal NorthRed     : std_logic;
    signal NorthYellow  : std_logic;
    signal NorthGreen   : std_logic;
    signal WestRed      : std_logic;
    signal WestYellow   : std_logic;
    signal WestGreen    : std_logic;

begin

    -- The Device Under Test (DUT)
    i_TrafficLights : entity work.T23_TrafficLights(rtl)
    generic map(ClockFrequencyHz => ClockFrequencyHz)
    port map (
        Clk         => Clk,
        nRst        => nRst,
        NorthRed    => NorthRed,
        NorthYellow => NorthYellow,
        NorthGreen  => NorthGreen,
        WestRed     => WestRed,
        WestYellow  => WestYellow,
        WestGreen   => WestGreen);
```

Then generating the clock with a ClockPeriod value and waiting for an edge to start-stop simulation. The simulation starts (nRst=1) when clk a rising edge is encountered in square-wave clock signal.

While nRst value=1 our traffic

light simulation acts like a flip flop continiously changing states.

```
-- Process for generating clock
Clk <= not Clk after ClockPeriod / 2;

-- Testbench sequence
process is
begin
    wait until rising_edge(Clk);
    wait until rising_edge(Clk);

    -- Take the DUT out of reset
    nRst <= '1';

    wait;
end process;
```

That was the testbench (main) code. The whole process actually runs inside the rtl (register-transfer level) file. So there is a procedure inside the first process which is sensitive to the clock signal which continiously updates the state and counter variables due to the ClockCycles variable declared inside of it.

```
begin

    process(Clk) is

        -- Procedure for changing state after a given time
        procedure ChangeState(ToState : t_State;
                              Minutes : integer := 0;
                              Seconds : integer := 0) is
            variable TotalSeconds : integer;
            variable ClockCycles  : integer;
        begin
            TotalSeconds := Seconds + Minutes * 60;
            ClockCycles  := TotalSeconds * ClockFrequencyHz -1;
            if Counter = ClockCycles then
                Counter <= 0;
                State   <= ToState;
            end if;
        end procedure;
```

But first, the variables/signals which will be used inside this rtl file must linked from the main file, so here's the portmap:

```
entity T23_TrafficLights is
generic(ClockFrequencyHz : integer);
port(
    Clk         : in std_logic;
    nRst        : in std_logic; -- Negative reset
    NorthRed    : out std_logic;
    NorthYellow : out std_logic;
    NorthGreen  : out std_logic;
    WestRed     : out std_logic;
    WestYellow  : out std_logic;
    WestGreen   : out std_logic);
end entity;
```

Lastly, we control each state with a long case-when statement due to state value if nRst!=0 , else resetting all values to the beggining states.

```
begin
    if rising_edge(Clk) then
        if nRst = '0' then
            -- Reset values
            State   <= NorthNext;
            Counter <= 0;
            NorthRed    <= '1';
            NorthYellow <= '0';
            NorthGreen  <= '0';
            WestRed     <= '1';
            WestYellow  <= '0';
            WestGreen   <= '0';
```

```
        else
            -- Default values
            NorthRed    <= '0';
            NorthYellow <= '0';
            NorthGreen  <= '0';
            WestRed     <= '0';
            WestYellow  <= '0';
            WestGreen   <= '0';

            Counter <= Counter + 1;

            case State is

                -- Red in all directions
                when NorthNext =>
                    NorthRed <= '1';
                    WestRed  <= '1';
                    ChangeState(StartNorth, Seconds => 5);

                -- Red and yellow in north/south direction
                when StartNorth =>
                    NorthRed    <= '1';
                    NorthYellow <= '1';
                    WestRed     <= '1';
                    ChangeState(North, Seconds => 5);

                -- Green in north/south direction
                when North =>
                    NorthGreen <= '1';
                    WestRed    <= '1';
                    ChangeState(StopNorth, Minutes => 1);
```

And here is the complete traffic lights implementation in vhdl.

Ideas of Improvement:

Adding thermal sensors to each light to calculate the crowd that is waiting at the time in the lights and generating a more suitable clockroutine with that information might be useful.

The number of traffic lights that need to be synchronized to improve the simulation can be increased by choosing more complex road designs. In addition, the current system reflects an ordinary daily life traffic light system, we can make this situation more active by making use of computer software, for example, the duration of the lights can be adjusted according to morning, noon and evening conditions, and in this way, more comfortable flow can be achieved in traffic at different times of the day. Apart from this, the duration of the traffic lights can be adjusted according to the vehicle density by looking at the map from a wider angle and seeing the vehicles coming and going from afar.