VHDL ASSIGNMENT

Introduction:

This assignment illustrates a VHDL (Very High-Speed Integrated Circuit Hardware Description Language) implementation of a pedestrian crossing the road project. This project emphasizes in the design of combinational and sequential digital circuits by using the VHDL. Furthermore, this report displays the use of an appropriate VHDL compiler and simulator. This report deals with the practical issues in the design of digital systems and it shows the uses of a FPGA (Field Programmable Gate Arrays) development board to implement digital logic designs.

Splitting of tasks:

All the tasks were made by one student and not a group. The process to construct this project was difficult to make by only one person. However, this project started with the design creation and it continued with the compile of this design. To check that the design is working right the simulator of Quartus Prime program used. After these two steps a hardware setup made to verify the code of the design by using the DE0-Nano FPGA board, a protoboard, six Led(2 yellow, 2 Red, 2 Green Led) and lots of female-male connectors.

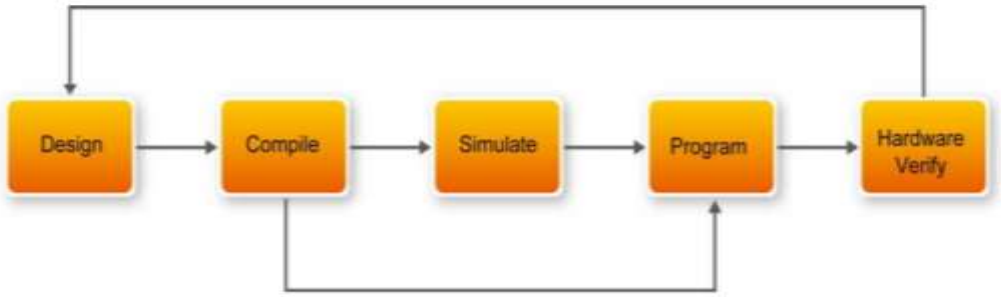


Figure 1 Design Flow

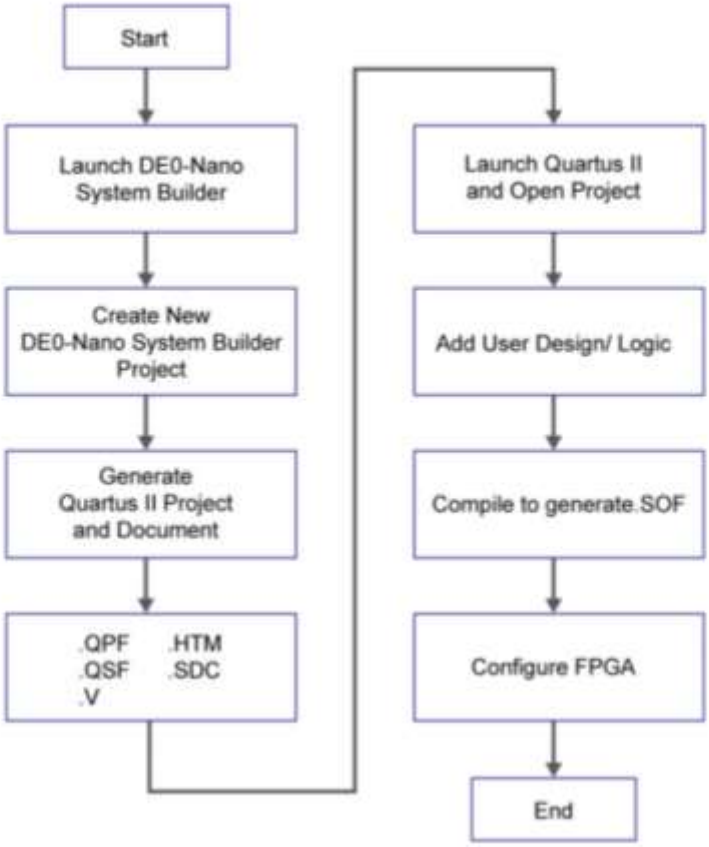


Figure 2 The general design flow of building a design

VHDL Code:

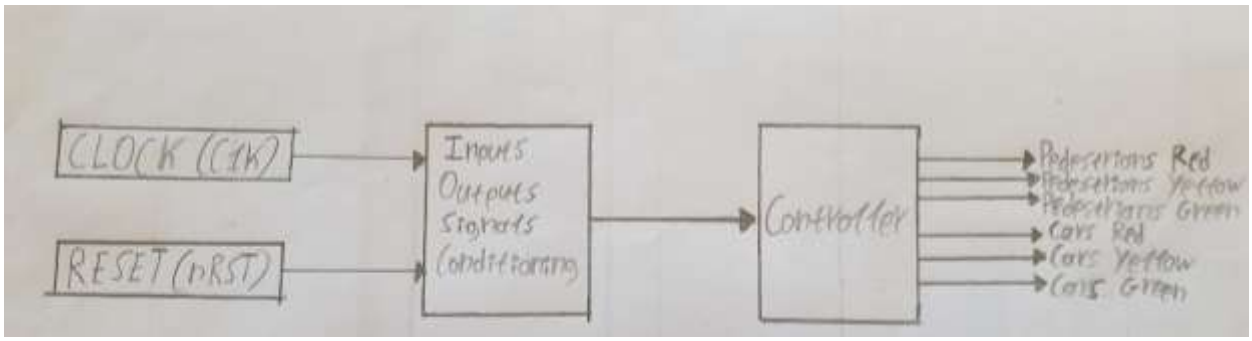


Figure 3 Block Diagram

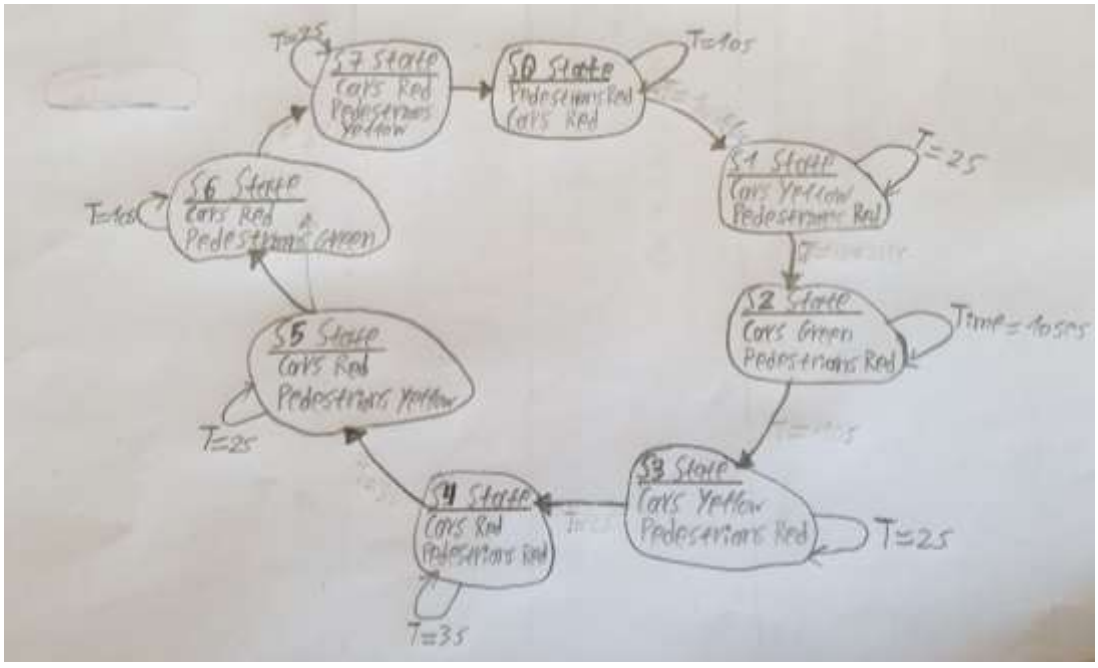


Figure 4 Structure Diagram

This design used to make the VHDL code specifications easier. Also, this design is an imitation of an FSM(Finite state machines) design, so this project works like a FSM system. The finite-state machine can be implemented using the state signal in a method with a Case statement. The statement of case contains a statement of when for each of the possible states which causes the program to take different paths for each state. Even the When statement can contain code that should be executed while in that state. Typically, when a predefined condition is met, the state must change.

VHDL Code Layout:

```
-- VHDL implementation of a
-- Pedestrian crossing project

-- Title: TrafficLights.vhd
--Designer
--Date
--Version No:1
--Target DE0 board - EP4CE22F17C6


library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity TrafficLights is
generic(ClockFrequencyHz:integer:=5000000);
port(
Clk: in std_logic;
nRst: in std_logic;      -- Reset
PedestriansRed : out std_logic;
PedestriansYellow : out std_logic;
PedestriansGreen: out std_logic;
CarsRed : out std_logic;
CarsYellow: out std_logic;
CarsGreen : out std_logic);
end TrafficLights;
```

```
architecture module of TrafficLights is
-- Enumerated type declaration and state signal declaration
type t_State is (s0, s1, s2, s3, s4, s5, s6, s7);


signal State : t_State;

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


begin

process(Clk) is

begin

if rising_edge(Clk) then

if nRst='1' then
-- Reset values

State <= s0;

Counter <= 0;

PedestriansRed    <= '0';

PedestriansYellow <= '1';

PedestriansGreen  <= '1';

CarsRed          <= '0';

CarsYellow       <= '1';

CarsGreen        <= '1';
```

```
else
-- Default values
Counter <= 0;

PedestriansRed    <= '0';

PedestriansYellow <= '1';

PedestriansGreen  <= '1';

CarsRed         <= '0';

CarsYellow       <= '1';

CarsGreen        <= '1';


Counter <= Counter +1;


case State is
-- Red in all directions
when s0 =>

PedestriansRed    <= '0';

PedestriansYellow <= '1';

PedestriansGreen  <= '1';

CarsRed         <= '0';

CarsYellow       <= '1';

CarsGreen        <= '1';

if Counter = ClockFrequencyHz *100 -1 then

Counter <=0;

State <= s1;

end if;
```

```
-- Red and yellow in Pedestrians/Cars direction
when s1 =>
PedestriansRed   <= '0';
PedestriansYellow <= '1';
PedestriansGreen  <= '1';
CarsRed         <= '1';
CarsYellow       <= '0';
CarsGreen        <= '1';
if Counter = ClockFrequencyHz * 20 -1  then
Counter <=0;
State <= s2;
end if;

-- Green/Red in Cars/Pedestrians direction
when s2  =>
PedestriansRed   <= '0';
PedestriansYellow <= '1';
PedestriansGreen  <= '1';
CarsRed         <= '1';
CarsYellow       <= '1';
CarsGreen        <= '0';
if Counter = ClockFrequencyHz * 100 -1 then
Counter <=0;
State <= s3;
end if;
```

```
-- Yellow/Red in Cars/Pedestrians direction
when s3 =>

PedestriansRed    <= '0';

PedestriansYellow <= '1';

PedestriansGreen  <= '1';

CarsRed         <= '1';

CarsYellow       <= '0';

CarsGreen        <= '1';

if Counter = ClockFrequencyHz * 20 -1  then

Counter <=0;

State <= s4;

end if;


-- Red in all directions
when s4 =>

PedestriansRed    <= '0';

PedestriansYellow <= '1';

PedestriansGreen  <= '1';

CarsRed         <= '0';

CarsYellow       <= '1';

CarsGreen        <= '1';

if Counter = ClockFrequencyHz * 40 -1 then

Counter <=0;

State <= s5;

end if;
```

```
 -- Red and yellow in Pedestrians/Cars direction
when s5 =>
PedestriansRed    <= '1';
PedestriansYellow <= '0';
PedestriansGreen  <= '1';
CarsRed         <= '0';
CarsYellow       <= '1';
CarsGreen        <= '1';
if Counter = ClockFrequencyHz * 20 -1 then
Counter <=0;
State <= s6;
end if;



-- Green/Red in Pedestrians/Cars direction
when s6 =>
PedestriansRed    <= '1';
PedestriansYellow <= '1';
PedestriansGreen  <= '0';
CarsRed         <= '0';
CarsYellow       <= '1';
CarsGreen        <= '1';
if Counter = ClockFrequencyHz * 100 -1  then
Counter <=0;
State <= s7;
end if;
```

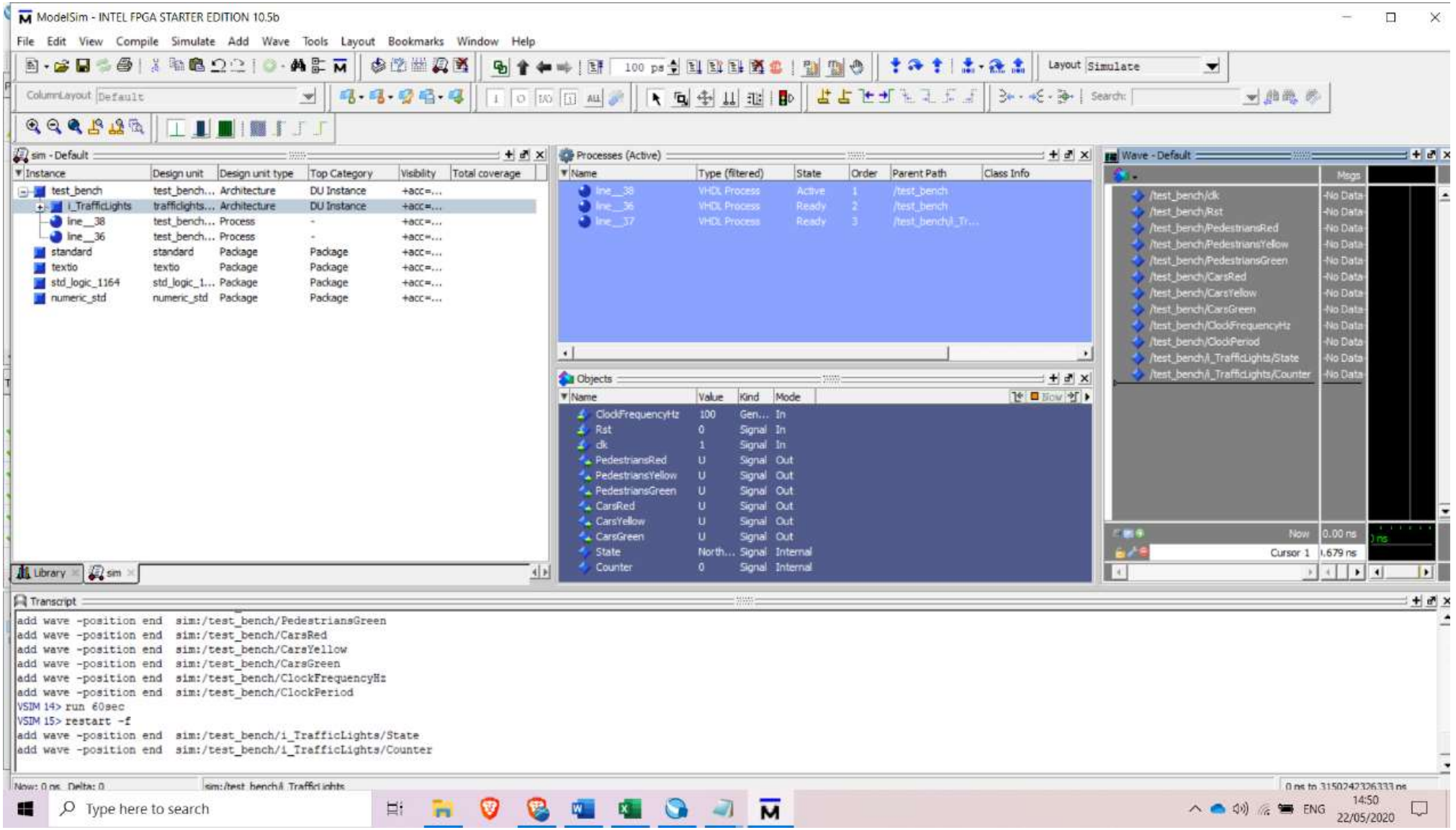
```
-- Yellow/Red in Pedestrians/Cars direction
when s7 =>

PedestriansRed    <= '1';

PedestriansYellow <= '0';

PedestriansGreen  <= '1';

CarsRed          <= '0';

CarsYellow       <= '1';

CarsGreen        <= '1';

if Counter = ClockFrequencyHz * 20 -1  then

Counter <=0;

State <= s0;

end if;

end case;


end if;

end if;

end process;

end architecture;
```

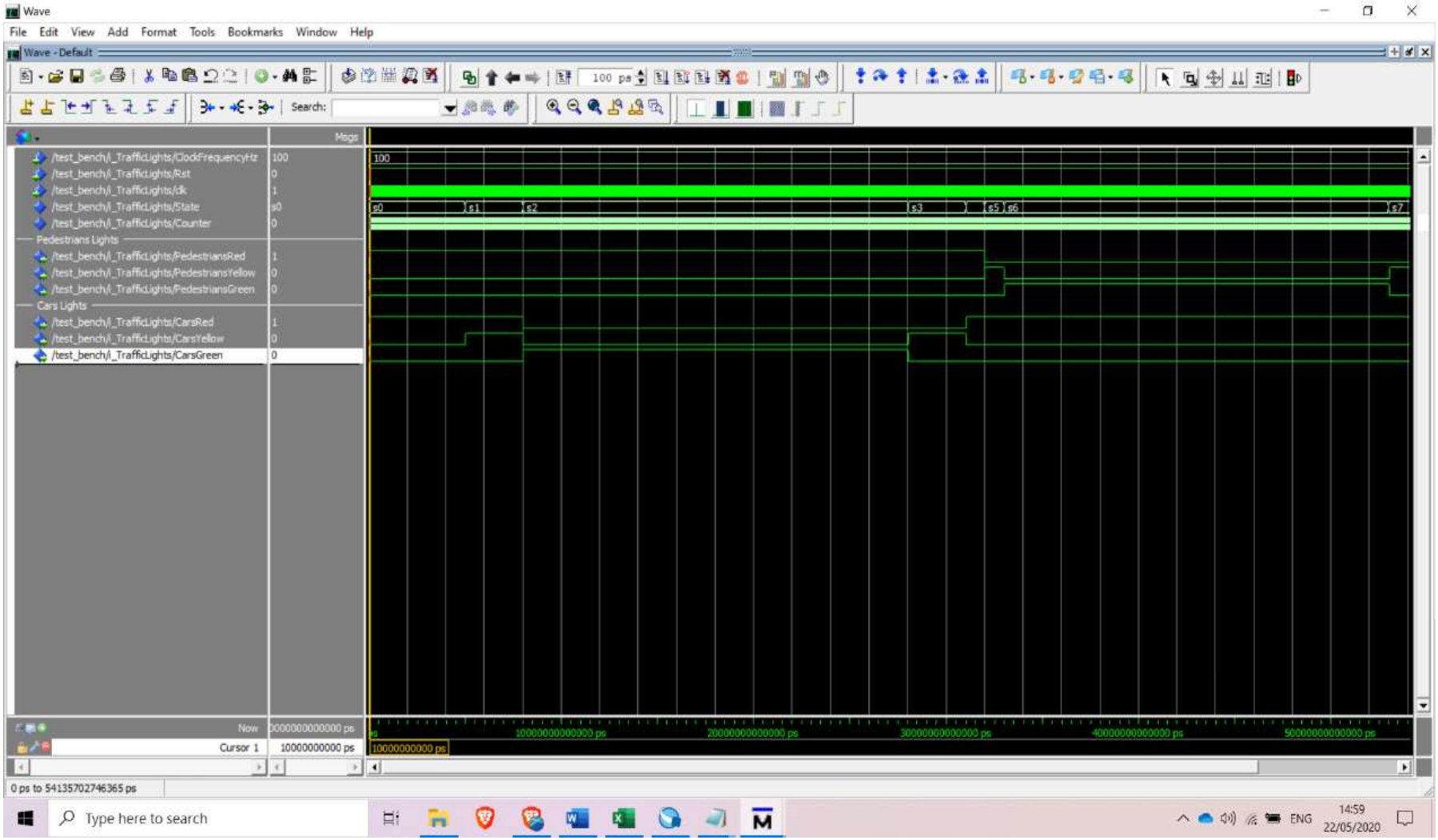
VHDL Simulation

Testbench Code:

```
use ieee.numeric_std.all;

entity test_bench is
end test_bench;

architecture test of test_bench is

constant ClockFrequencyHz : integer := 100;
constant ClockPeriod      : time    := 1000 ms / ClockFrequencyHz ;

signal Clk:              std_logic := '1' ;
signal nRst:             std_logic := '0' ;
signal PedestriansRed:   std_logic;
signal PedestriansYellow: std_logic;
signal PedestriansGreen:  std_logic;
signal CarsRed:          std_logic;
signal CarsYellow:       std_logic;
signal CarsGreen:        std_logic;

begin
i_TrafficLights : entity work.TrafficLights
generic map(ClockFrequencyHz => ClockFrequencyHz)
port map (
Clk => Clk,
nRst => nRst,
PedestriansRed    => PedestriansRed,
PedestriansYellow => PedestriansYellow,
```

```
PedestriansGreen  => PedestriansGreen,
CarsRed          => CarsRed,
CarsYellow       => CarsYellow,
CarsGreen        => CarsGreen);

Clk <= not Clk after ClockPeriod /2;

process is
begin
wait until rising_edge(Clk);
wait until rising_edge(Clk);

nRst <= '1';

wait;
end process;

end architecture;
```


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
sim - Default
Instance Design unit Design unit type Top Category Visibility Total coverage
test_bench test_bench... Architecture DU Instance +acc=...
i_TrafficLights trafficlights... Architecture DU Instance +acc=...
line__38 test_bench... Process - +acc=...
line__36 test_bench... Process - +acc=...
standard standard Package Package +acc=...
textio textio Package Package +acc=...
std_logic_1164 std_logic_1... Package Package +acc=...
numeric_std numeric_std Package Package +acc=...
Processes (Active)
Name Type (filtered) State Order Parent Path Class Info
line__38 VHDL Process Active 1 /test_bench
line__36 VHDL Process Ready 2 /test_bench
line__37 VHDL Process Ready 3 /test_bench/i_Tr...
Objects
Name Value Kind Mode
ClockFrequencyHz 100 Gen... In
Rst 0 Signal In
clk 1 Signal In
PedestriansRed U Signal Out
PedestriansYellow U Signal Out
PedestriansGreen U Signal Out
CarsRed U Signal Out
CarsYellow U Signal Out
CarsGreen U Signal Out
State North... Signal Internal
Counter 0 Signal Internal
Wave - Default
Msgs
/test_bench/clk
/test_bench/Rst
/test_bench/PedestriansRed
/test_bench/PedestriansYellow
/test_bench/PedestriansGreen
/test_bench/CarsRed
/test_bench/CarsYellow
/test_bench/CarsGreen
/test_bench/ClockFrequencyHz
/test_bench/ClockPeriod
/test_bench/i_TrafficLights/State
/test_bench/i_TrafficLights/Counter
No Data
Now 0.00 ns
Cursor 1 1.679 ns
Transcript
add wave -position end sim:/test_bench/PedestriansGreen
add wave -position end sim:/test_bench/CarsRed
add wave -position end sim:/test_bench/CarsYellow
add wave -position end sim:/test_bench/CarsGreen
add wave -position end sim:/test_bench/ClockFrequencyHz
add wave -position end sim:/test_bench/ClockPeriod
VSIM 14> run 60sec
VSIM 15> restart -f
add wave -position end sim:/test_bench/i_TrafficLights/State
add wave -position end sim:/test_bench/i_TrafficLights/Counter
Type here to search
ENG
14:50
22/05/2020


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/test_bench/i_TrafficLights/ClockFrequencyHz
/test_bench/i_TrafficLights/Rst
/test_bench/i_TrafficLights/clk
/test_bench/i_TrafficLights/State
/test_bench/i_TrafficLights/Counter
Pedestrians Lights
/test_bench/i_TrafficLights/PedestriansRed
/test_bench/i_TrafficLights/PedestriansYellow
/test_bench/i_TrafficLights/PedestriansGreen
Cars Lights
/test_bench/i_TrafficLights/CarsRed
/test_bench/i_TrafficLights/CarsYellow
/test_bench/i_TrafficLights/CarsGreen
100
0
1
s0
0
1
0
0
1
0
0
s0 s1 s2 s3 s5 s6 s7
Now
Cursor 1
10000000000 ps
100000000000 ps
200000000000 ps
300000000000 ps
400000000000 ps
500000000000 ps
0 ps to 54135702746365 ps
Type here to search
ENG
14:59
22/05/2020

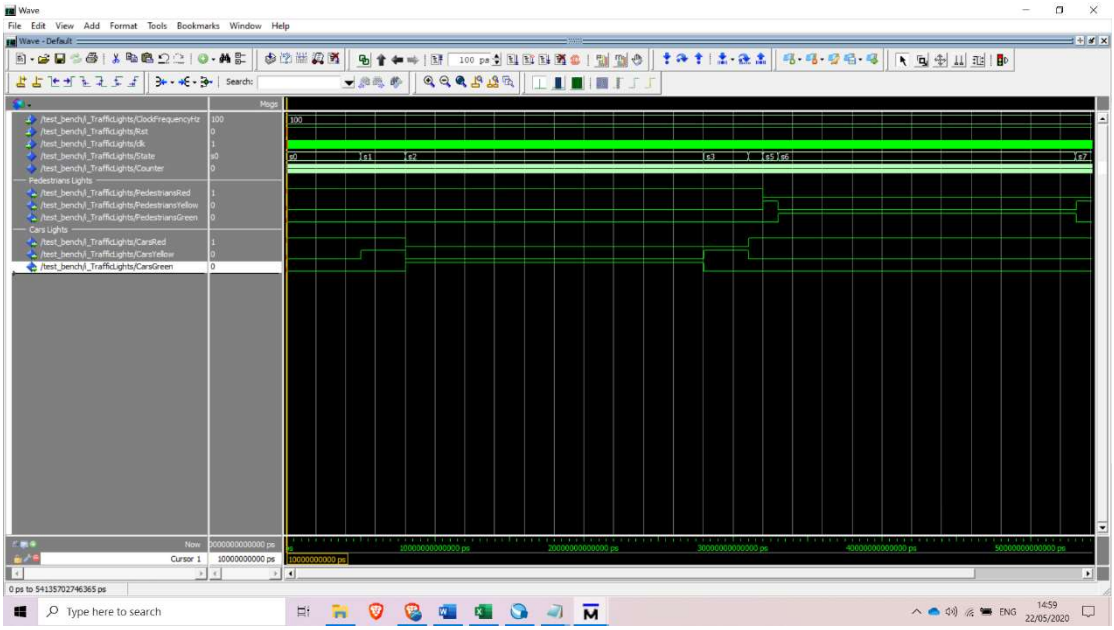
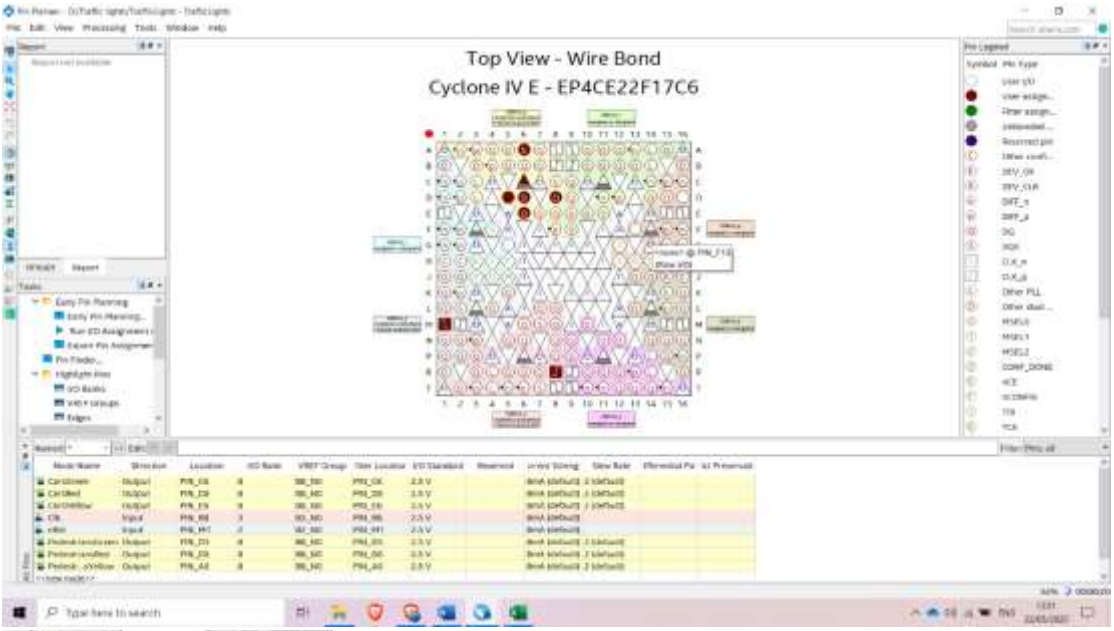


Figure 5,6,7 Simulation Results

The simulation results show how the FSM works and the delays between them. Also, it displays the frequency, the reset(Rst), the clock(clk), the state and the Counter. From the simulation images, the state change from the s0 to s7 and then goes back again in a specific time and the frequency has changed only for practice purposes from 5000000 to 100Hz. The clock and the reset works correctly.

Testing of Code:



Node Name	Direction	Location	I/O Bank	VREF Group	...tter Locatior	I/O Standard	Reserved	...rrent Streng	Slew Rate
CarsGreen	Output	PIN_C6	8	B8_N0	PIN_C6	2.5 V		8mA (default)	2 (default)
CarsRed	Output	PIN_D8	8	B8_N0	PIN_D8	2.5 V		8mA (default)	2 (default)
CarsYellow	Output	PIN_E6	8	B8_N0	PIN_E6	2.5 V		8mA (default)	2 (default)
Clk	Input	PIN_R8	3	B3_N0	PIN_R8	2.5 V		8mA (default)	
nRst	Input	PIN_M1	2	B2_N0	PIN_M1	2.5 V		8mA (default)	
PedestriansGreen	Output	PIN_D5	8	B8_N0	PIN_D5	2.5 V		8mA (default)	2 (default)
PedestriansRed	Output	PIN_D6	8	B8_N0	PIN_D6	2.5 V		8mA (default)	2 (default)
Pedestr...sYellow	Output	PIN_A6	8	B8_N0	PIN_A6	2.5 V		8mA (default)	2 (default)

Figure 8 Pin Allocations

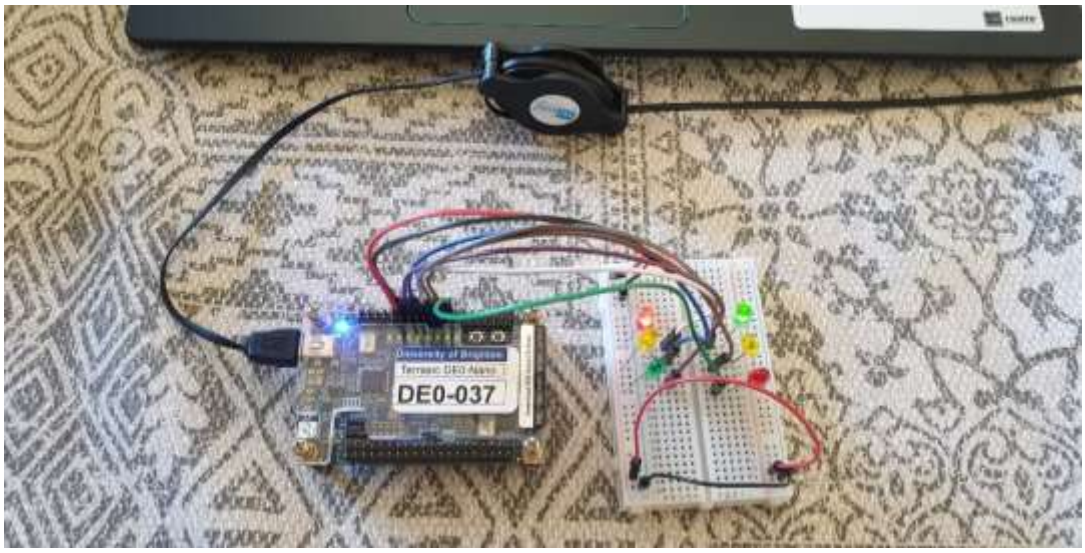


Figure 9 Target Board

The results from these connections are the led work properly. The Pin allocations, led connections were found from the DE0-Nano board Manual. The LEDs are slotted to the VCC. This protoboard takes from the de0 board all the voltage, current that it needs by using specific connections which also found in the manual.

Conclusion:

To conclude, the project worked well in steps during the semester. There were some issues with the delays between the changing states, but they solved by searching through the construction of the VHDL Code and studying further the lectures of this module. Furthermore, another problem that came up was the hardware setup(protoboard connection with de0 board) which also solved by reading the manual of the de0 board. This module was challenging because of the work that the student had to do. To understand fully this module a student needed to work a lot through the semester on his own time to achieve this and this is why this project was challenging.