**DLDA**

**MINI-PROJECT REPORT**

PROBLEM STATEMENT:

ONE WAY TRAFFIC CONTROLLER USING VHDL

PROJECT DONE BY

SHUBHAM BHATE-8318

PRINCETON BARETTO-8316

**STATE DIAGRAM**

STATE 1: RED







STATE 3: YELLOW STATE 2:GREEN

**STATE TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| **STATE** | **RED** | **GREEN** | **YELLOW** |
| **1** | **1** | **0** | **0** |
| **2** | **0** | **1** | **0** |
| **3** | **0** | **0** | **1** |

**CIRCUIT DIAGRAM**



PRE



DR  QR  DG QG  DY  QY 







QR QG QY







A 3-bit Ring Counter is used to implement a

One way traffic controller. 3 D flip flops are used ,one for each color. The output of one flip flop is connected to the input of the next flip flop. The output of the last flip flop is given to the input of the first flip flop. Clock is given to all flip flops. Initially the Red flip flop is given preset input while the rest are cleared. After 5 counts the signal changes to Green .Again after 5 more counts the signal changes to Yellow for 2 counts before returning back to the initial state.

**VHDL CODE :**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating

---- any Xilinx primitives in this code.

---library UNISIM;

--use UNISIM.VComponents.all;

entity tfcontroller\_code is

    Port ( clk : in  STD\_LOGIC;

           rst : in  STD\_LOGIC;

           Green : out  STD\_LOGIC;

           Red : out  STD\_LOGIC;

           Yellow : out  STD\_LOGIC);

end tfcontroller\_code;

architecture Behavioral of tfcontroller\_code is

signal count:integer range 0 to 10 := 0;

signal state:integer range 0 to 2 := 0;

begin

    process(clk, rst)

    begin

        if(rst = '1') then

            state <= 0;

            Red <= '1';

            Green <= '0';

            Yellow <= '0';

            count <= 0;

        elsif clk'event and clk='1' then

        case state is

        when 0 =>  --Red Light

        if(count=5) then

            count <= 0;

            state <= 1;

        else

            count <= (count + 1);

            Red <= '1';

            Green <= '0';

            Yellow <= '0';

        end if;

        when 1 =>  --Green Light

        if(count=5) then

            count <= 0;

            state <= 2;

        else

            count <= count + 1;

            Red <= '0';

            Green <= '1';

            Yellow <= '0';

        end if;

        when 2 =>  --Yellow Light

        if(count=2) then

            count <= 0;

            state <= 0;

        else

            count <= count + 1;

            Red <= '0';

            Green <= '0';

            Yellow <= '1';

        end if;

        when others =>

            state <= 0;

            count <= 0;

        end case;

        end if;

    end process;

end Behavioral;

**OUTPUT :**

