Université Abdelmalek Essaâdi
École Nationale des Sciences Appliquées de Tétouan

Module : Electronique embarquée
Filière : GM2

Rapports du mini-projet

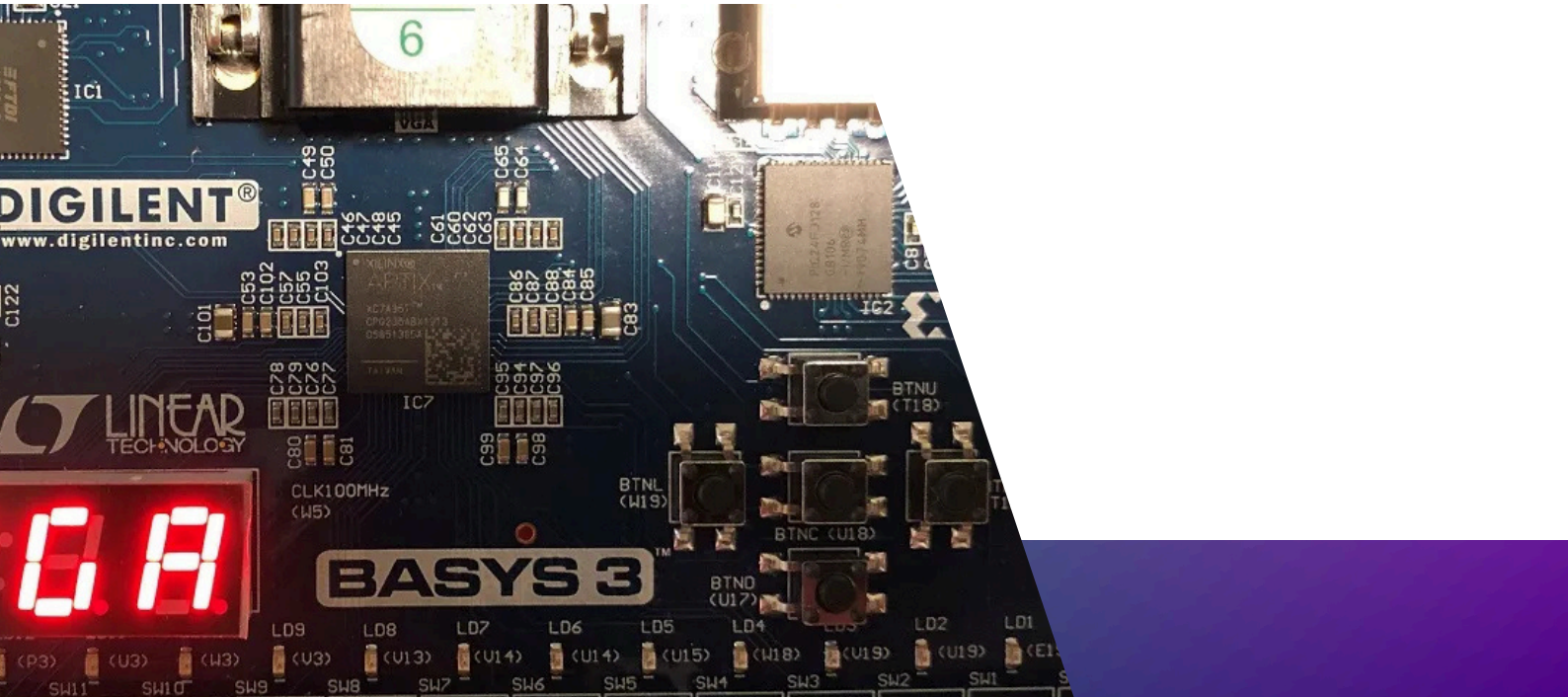
Conception d'un système qui contrôle des feux tri-couleurs d'un carrefour

Réalisé par :

- Wissal MTAHTAH

2024/2025

Sommaire



Introduction

Dans le cadre de ce projet, nous avons été amenés à concevoir un système de contrôle de feux tri-couleurs (rouge, vert, orange) d'un carrefour, en utilisant le langage VHDL et les machines à états finis (FSM).
Les systèmes de feux tricolores sont essentiels pour la régulation du trafic et la sécurité routière. Ils doivent respecter des cycles précis et être capables de réagir à des conditions spécifiques (comme un redémarrage après un reset).
Le choix du VHDL (VHSIC Hardware Description Language) s'impose naturellement car il s'agit d'un langage dédié à la description et la simulation des circuits logiques numériques. Grâce à lui, nous pouvons modéliser des comportements séquentiels complexes, comme les transitions entre états d'un feu de circulation, tout en assurant une grande précision et une implémentation potentielle sur FPGA.
Dans ce rapport, nous détaillons le travail réalisé, en expliquant d'abord le fonctionnement général, puis en présentant les schémas d'états, les codes VHDL, ainsi que les résultats des simulations sous Vivado.

Pourquoi on utilise VHDL

Le choix du langage VHDL est motivé par plusieurs raisons :

- Il permet une description précise et standardisée des comportements matériels numériques.
- Il est largement utilisé dans l'industrie pour le design FPGA et ASIC.
- Il offre la possibilité de simuler les comportements avant implémentation réelle, ce qui est essentiel pour valider les machines à états finis.
- Son approche structurée permet de modulariser et réutiliser les composants facilement, même pour des projets plus complexes.

Pourquoi on utilise VIVADO

Vivado est un environnement de développement intégré (IDE) puissant développé par Xilinx, spécifiquement conçu pour la conception, la simulation et la synthèse des circuits numériques destinés aux FPGA. Il offre une interface conviviale pour écrire, analyser et vérifier les codes VHDL, ainsi que des outils avancés de simulation permettant d'observer le comportement des signaux en temps réel. Vivado permet également d'optimiser l'utilisation des ressources matérielles et de générer des bitstreams pour le déploiement sur des cartes FPGA. Pour ce projet, Vivado a été choisi car il offre une compatibilité parfaite avec les langages HDL (comme VHDL), facilite la simulation des machines à états finis, et fournit un environnement complet allant de la conception à l'implémentation.

Description du sujet

Le système que nous avons conçu repose sur une machine à états finis comportant trois états principaux :

- Rouge (RED) : Le feu rouge est allumé, les autres sont éteints, pendant une durée définie.
- Vert (GREEN) : Le feu vert est allumé, permettant aux véhicules de passer.
- Orange (ORANGE) : Le feu orange avertit du changement imminent, pendant une courte durée avant de revenir au rouge.

Le module est cadencé par une horloge (clk) et peut être réinitialisé (reset) à tout moment.

Nous avons décrit ce comportement dans un fichier VHDL en définissant d'abord les entrées et sorties, puis en créant un processus séquentiel qui gère les transitions d'états en fonction d'un compteur.

Parallèlement, nous avons développé un banc de test (testbench) qui génère un signal d'horloge et applique des stimuli au système pour observer son comportement. Les résultats ont été visualisés sous forme de simulations dans Vivado, permettant de vérifier que les transitions respectaient bien le schéma attendu.

On a veillé à respecter l'ensemble des exigences : code VHDL inséré en format texte, graphe d'état fourni, et capture d'écran des résultats de simulation.

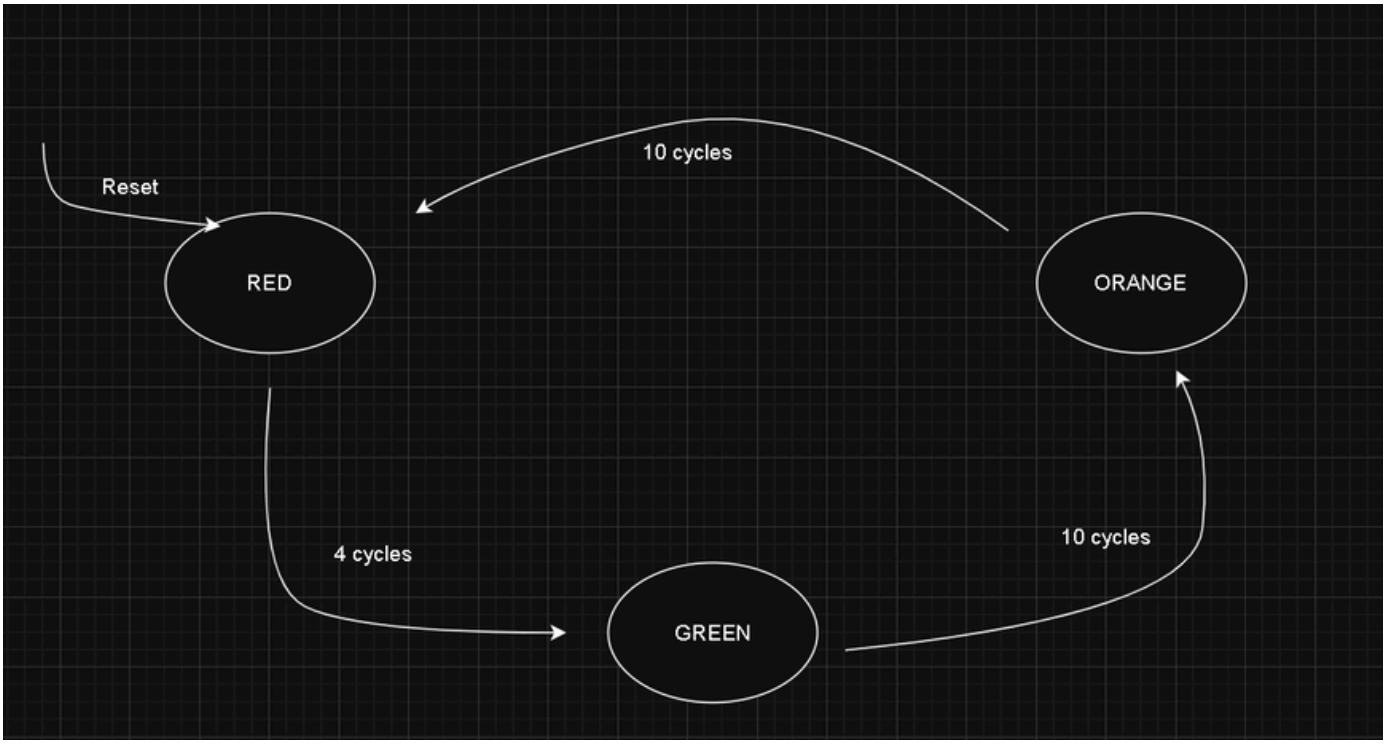
Graphe d’état

Le système fonctionne selon trois états principaux :

- RED : Le feu rouge est allumé, les autres sont éteints.
- GREEN : Le feu vert est allumé, les autres sont éteints.
- ORANGE : Le feu orange est allumé, les autres sont éteints.

Les transitions sont contrôlées par un compteur interne :

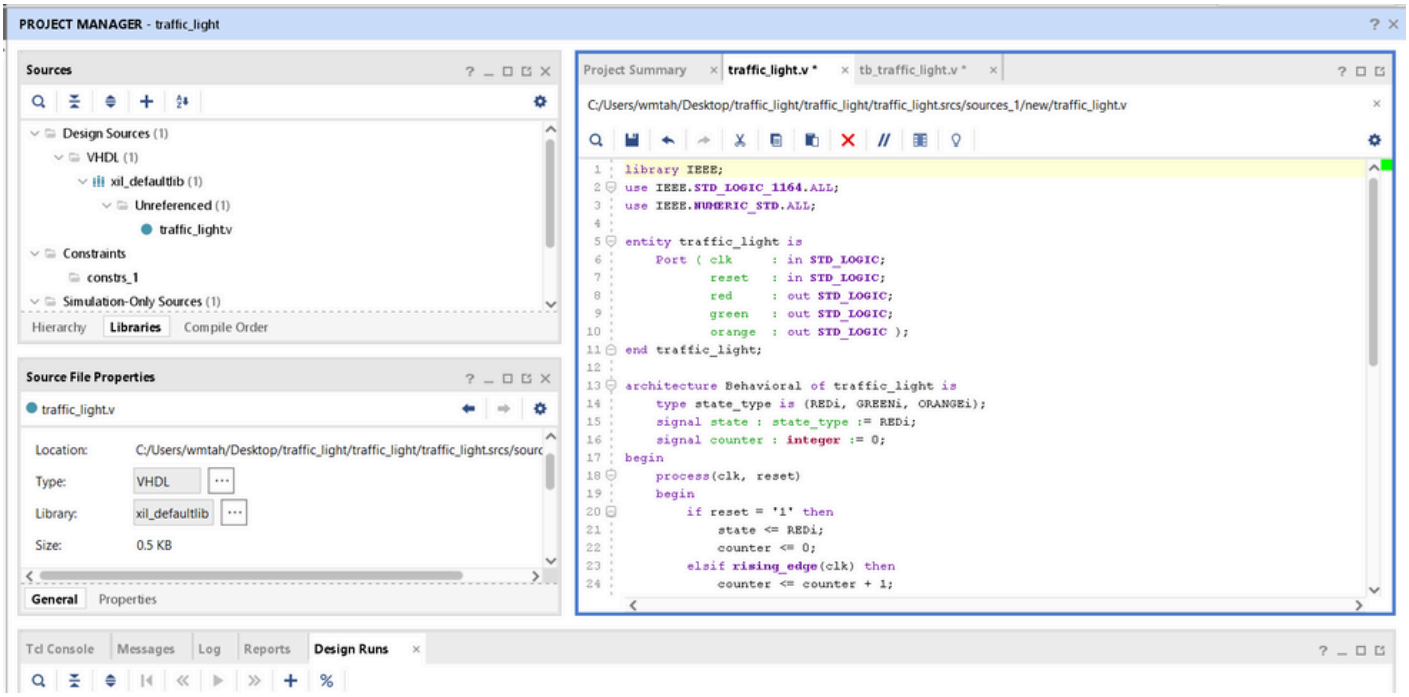
- RED → GREEN après 10 cycles.
- GREEN → ORANGE après 10 cycles.
- ORANGE → RED après 4 cycles.

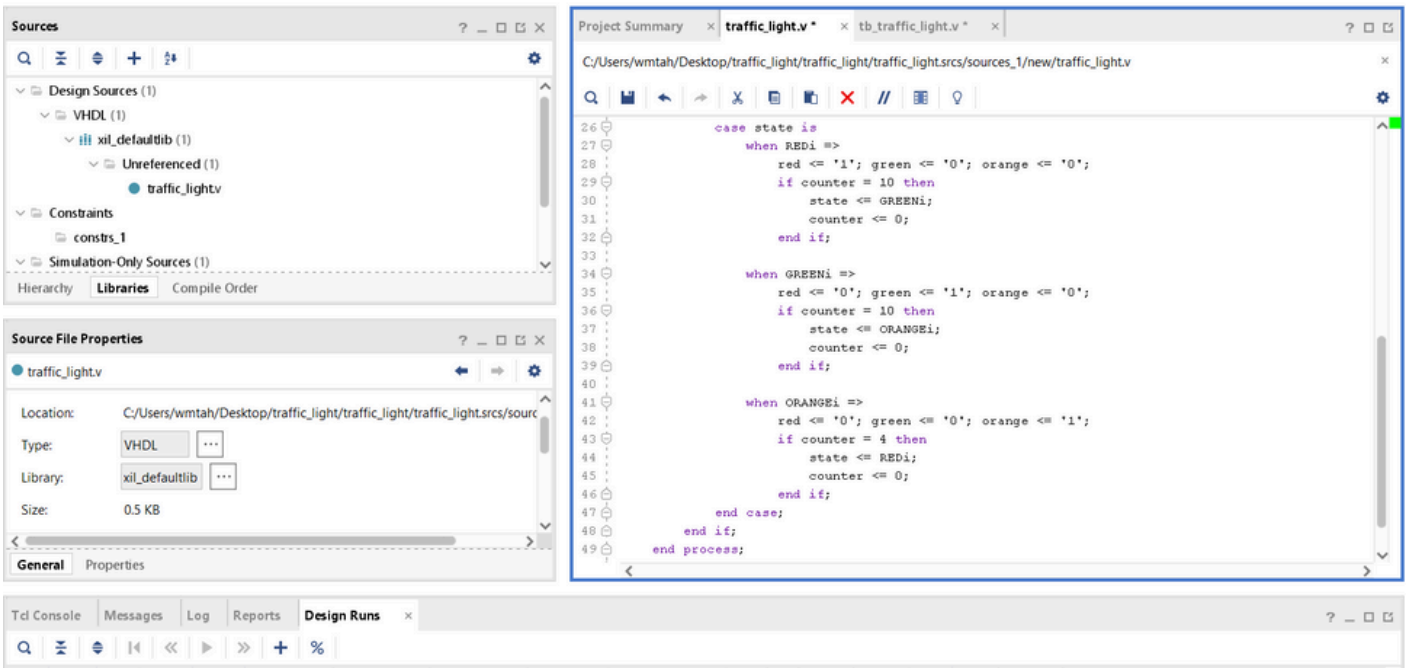


Code VHDL (traffic light controller)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity traffic_light is
    Port ( clk    : in STD_LOGIC;
           reset  : in STD_LOGIC;
           red    : out STD_LOGIC;
           green  : out STD_LOGIC;
           orange : out STD_LOGIC );
end traffic_light;

architecture Behavioral of traffic_light is
    type state_type is (REDi, GREENi, ORANGEi);
    signal state : state_type := REDi;
    signal counter : integer := 0;
begin
    process(clk, reset)
    begin
        if reset = '1' then
            state <= REDi;
            counter <= 0;
        elsif rising_edge(clk) then
            counter <= counter + 1;

            case state is
                when REDi =>
                    red <= '1'; green <= '0'; orange <= '0';
                    if counter = 10 then
                        state <= GREENi;
                        counter <= 0;
                    end if;

                when GREENi =>
                    red <= '0'; green <= '1'; orange <= '0';
                    if counter = 10 then
                        state <= ORANGEi;
                        counter <= 0;
                    end if;

                when ORANGEi =>
                    red <= '0'; green <= '0'; orange <= '1';
                    if counter = 4 then
                        state <= REDi;
                        counter <= 0;
                    end if;
            end case;
        end if;
    end process;
end Behavioral;
```

Code VHDL (traffic light controller)

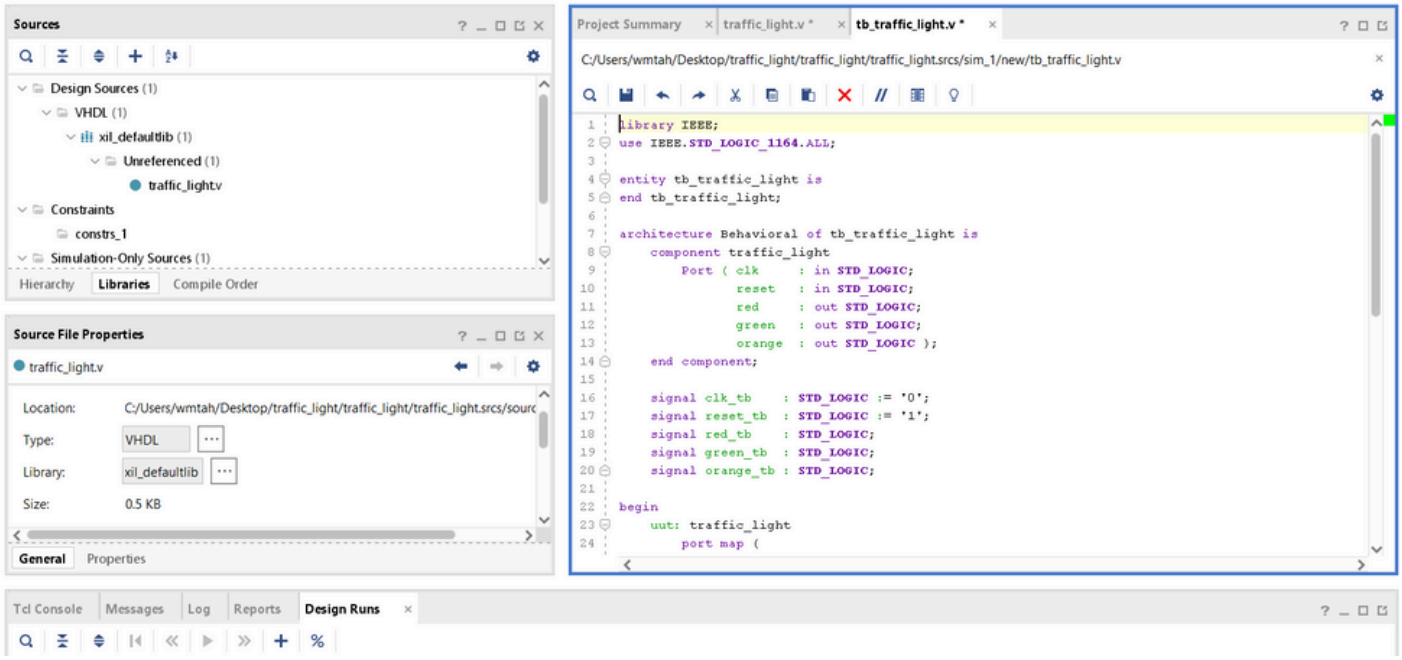


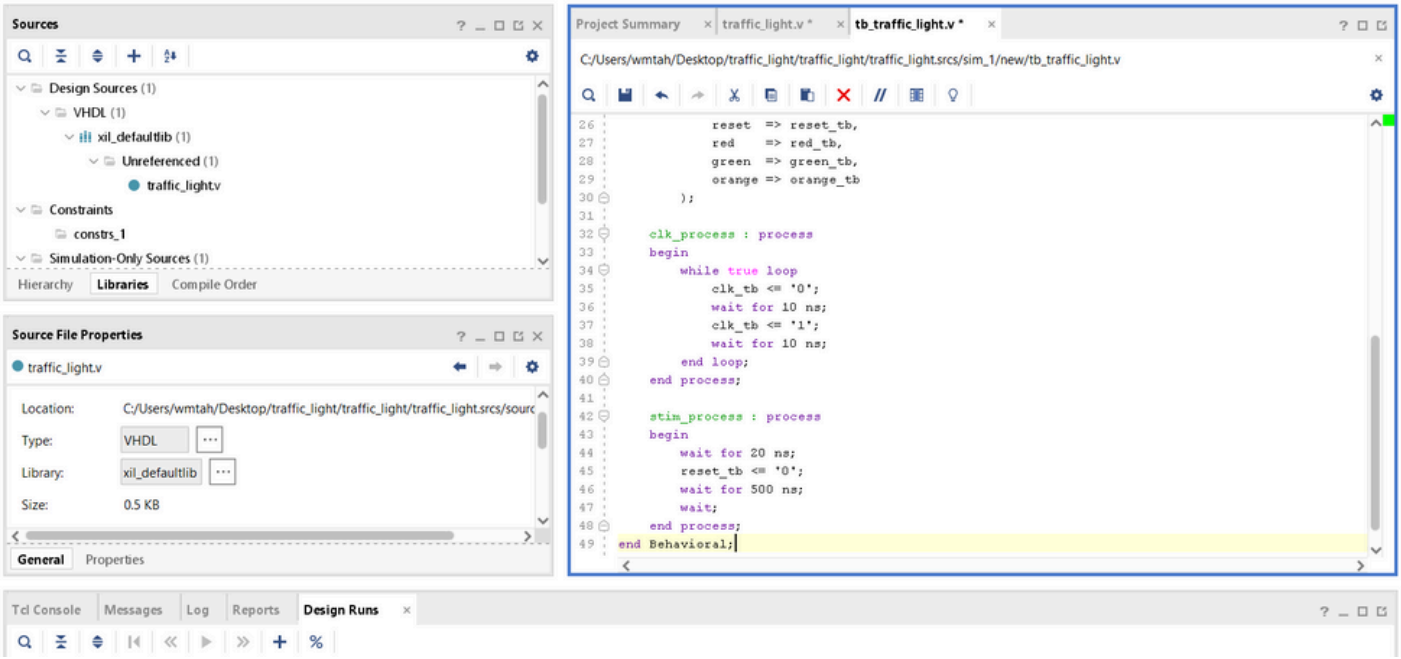


Testbench

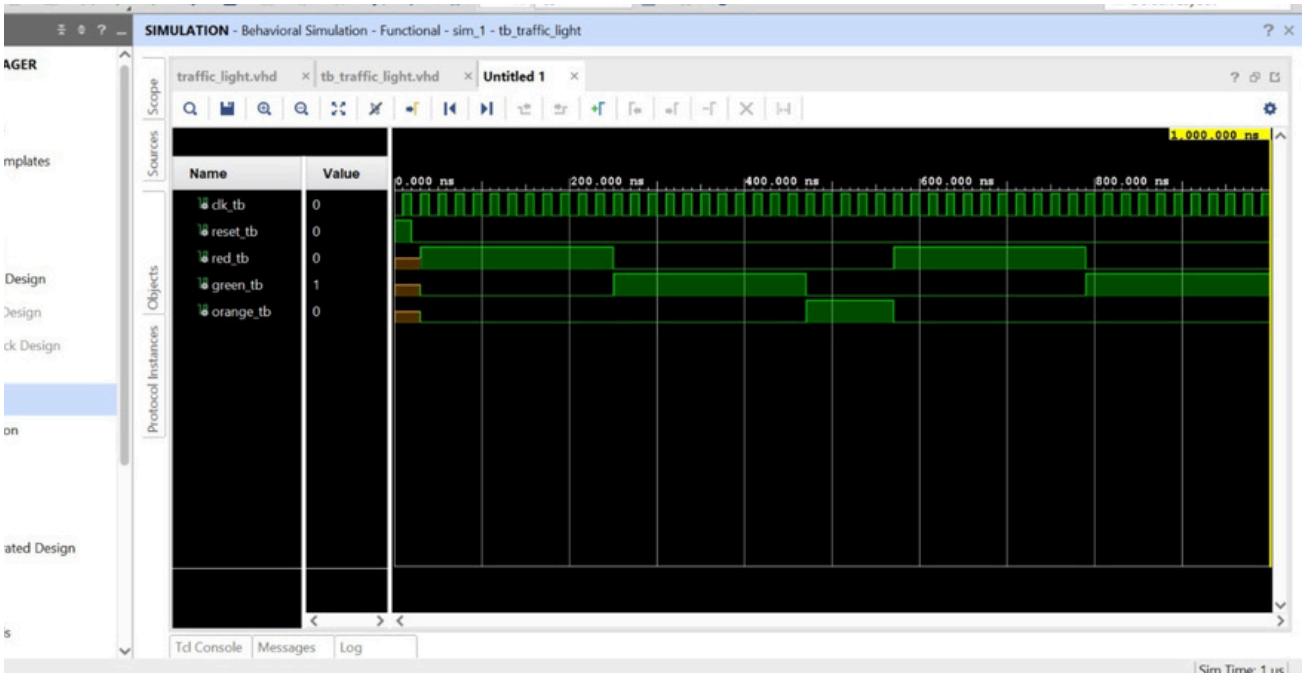
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_traffic_light is
end tb_traffic_light;

architecture Behavioral of tb_traffic_light is
   component traffic_light
      Port ( clk    : in STD_LOGIC;
             reset  : in STD_LOGIC;
             red    : out STD_LOGIC;
             green  : out STD_LOGIC;
             orange : out STD_LOGIC );
   end component;

   signal clk_tb    : STD_LOGIC := '0';
   signal reset_tb  : STD_LOGIC := '1';
   signal red_tb    : STD_LOGIC;
   signal green_tb  : STD_LOGIC;
   signal orange_tb : STD_LOGIC;

begin
   uut: traffic_light
      port map (
         clk    => clk_tb,
         reset  => reset_tb,
         red    => red_tb,
         green  => green_tb,
         orange => orange_tb
      );

   clk_process : process
   begin
      while true loop
         clk_tb <= '0';
         wait for 10 ns;
         clk_tb <= '1';
         wait for 10 ns;
      end loop;
   end process;

   stim_process : process
   begin
      wait for 20 ns;
      reset_tb <= '0';

      wait for 500 ns;

      wait;
   end process;

end Behavioral;
```

Testbench


Sources
Design Sources (1)
VHDL (1)
xil_defaultlib (1)
Unreferenced (1)
traffic_light.v
Constraints
constrs_1
Simulation-Only Sources (1)
Hierarchy
Libraries
Compile Order
Source File Properties
traffic_light.v
Location: C:/Users/wmtah/Desktop/traffic_light/traffic_light/traffic_light.srcs/sourc
Type: VHDL
Library: xil_defaultlib
Size: 0.5 KB
General
Properties
Project Summary
traffic_light.v *
tb_traffic_light.v *
C:/Users/wmtah/Desktop/traffic_light/traffic_light/traffic_light.srcs/sim_1/new/tb_traffic_light.v
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_traffic_light is
end tb_traffic_light;
architecture Behavioral of tb_traffic_light is
component traffic_light
Port (clk : in STD_LOGIC;
reset : in STD_LOGIC;
red : out STD_LOGIC;
green : out STD_LOGIC;
orange : out STD_LOGIC);
end component;
signal clk_tb : STD_LOGIC := '0';
signal reset_tb : STD_LOGIC := '1';
signal red_tb : STD_LOGIC;
signal green_tb : STD_LOGIC;
signal orange_tb : STD_LOGIC;
begin
uut: traffic_light
port map (
Tcl Console
Messages
Log
Reports
Design Runs


Sources
Design Sources (1)
VHDL (1)
xil_defaultlib (1)
Unreferenced (1)
traffic_light.v
Constraints
constrs_1
Simulation-Only Sources (1)
Hierarchy
Libraries
Compile Order
Source File Properties
traffic_light.v
Location: C:/Users/wmtah/Desktop/traffic_light/traffic_light/traffic_light.srcs/sourc
Type: VHDL
Library: xil_defaultlib
Size: 0.5 KB
General
Properties
Project Summary
traffic_light.v *
tb_traffic_light.v *
C:/Users/wmtah/Desktop/traffic_light/traffic_light/traffic_light.srcs/sim_1/new/tb_traffic_light.v
reset => reset_tb,
red => red_tb,
green => green_tb,
orange => orange_tb
);
clk_process : process
begin
while true loop
clk_tb <= '0';
wait for 10 ns;
clk_tb <= '1';
wait for 10 ns;
end loop;
end process;
stim_process : process
begin
wait for 20 ns;
reset_tb <= '0';
wait for 500 ns;
wait;
end process;
end Behavioral;
Tcl Console
Messages
Log
Reports
Design Runs

Simulation



La simulation affichée ci-dessus représente le comportement du système de feux tricolores modélisé en VHDL. On peut y observer :

- clk_tb : Le signal d'horloge utilisé pour synchroniser les changements d'état.
- reset_tb : Le signal de remise à zéro, qui force le système à revenir à l'état initial (rouge).
- red_tb, green_tb, orange_tb : Les signaux de sortie qui activent respectivement les feux rouge, vert et orange.

Dans cette simulation, on remarque clairement les cycles :

- D'abord, le feu rouge est actif pendant une période donnée (10 cycles),
- Ensuite, le feu passe au vert pour 10 cycles,
- Puis il passe à l'orange pour 4 cycles, avant de revenir au rouge.

Ces transitions permettent de vérifier que la machine à états finis fonctionne comme prévu et que chaque état respecte la durée programmée. Les signaux changent exactement au moment attendu, ce qui confirme que notre conception répond bien aux exigences fonctionnelles.

Conclusion

Ce projet a permis de mettre en pratique les concepts de machines à états finis et d'implémentation en VHDL pour concevoir un système réel et utile : le contrôleur de feux tricolores.
Grâce à la simulation sous Vivado, nous avons pu vérifier que notre conception répondait aux exigences, notamment en termes de séquence d'allumage et de transitions entre états.
Cette expérience nous a permis de mieux comprendre les défis liés à la conception numérique, d'améliorer nos compétences en VHDL et de développer une approche rigoureuse pour aborder des systèmes séquentiels. Elle ouvre la voie à des projets plus avancés, comme la prise en compte de capteurs ou la gestion multi-voies, qui pourraient enrichir ce système à l'avenir.