MAINGUET Alice
PHAM Nu-Huyen-Trang

MI01 - Compte rendu TP3

Exercice 1 - Test de diviseur de fréquence

Q1.

sur PC

Q2.

On a mis les commentaires sur le fonctionnement du diviseur directement sur le code :

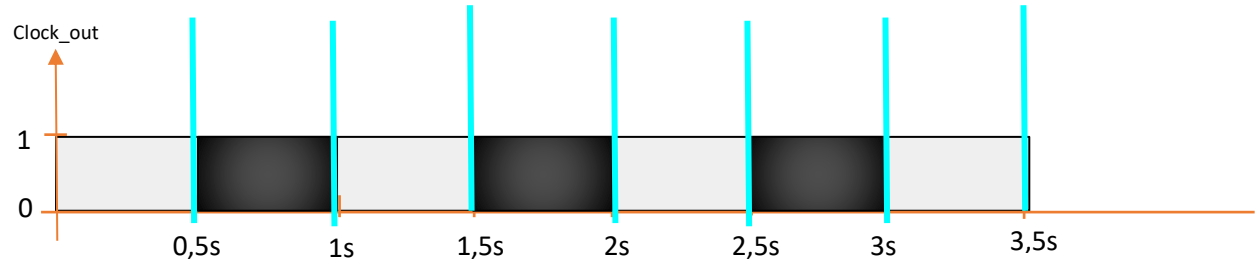
```
entity clock_divider is
        generic(divisor : integer := 100000000); -- Valeur du diviseur. La fréquence de sortie en dépend.
        port(clock_in, reset : in bit; --clock_in: Horloge à la fréquence initiale
             clock_out : out bit); --Horloge modifiée à la fréquence voulue grâce au diviseur de fréquence
end clock_divider;

architecture Behavioral of clock_divider is
begin
process(clock_in, reset)
 variable c : integer range 0 to divisor - 1 := 0;
-- On introduit une variable c
-- c permet de parcourir un cycle de divisor (c parcourt toutes les valeurs entre 0 et divisor-1)
        begin
        if reset = '1' then c := 0; --Réinitialisation du cycle en cas d'appui sur Reset
        elsif clock_in'event and clock_in = '1' then
                if c < (divisor - 1) then c := c + 1;
         --c prend successivement les valeurs de 0 à divisor-1 à chaque front montant(tant qu'on n'active pas reset)
                else c := 0;  --arrivés à divisor-1: on redémarre le cycle à 0
                end if;
        end if.


        --Ici 100 MHz = 100 000 000 Hz = divisor, donc un cycle(c parcourt toutes les valeurs de divisor) dure une seconde
        if c < (divisor - 1) / 2 then clock_out <= '0';
        --Sur la première moitié du cycle (première demi-seconde), on défini la sortie à 0
           else clock_out <= '1';
        --Sur la seconde moitié du cycle (deuxième demi-seconde), on défini la sortie à 1
        end if;
        end process;

end Behavioral;
```

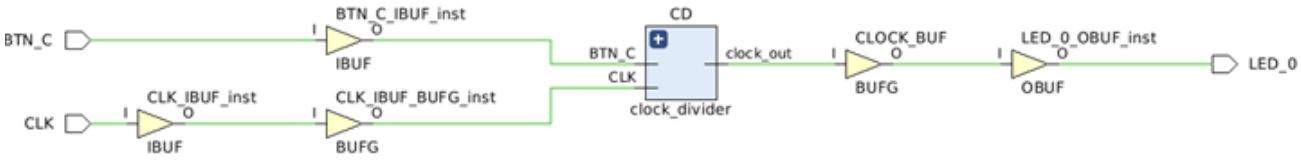
Chronnogramme du comportement de clock_out (pour divisor = 100 000 000) :



Q3.

Dans l'exemple, la fréquence d'entrée est toujours 100 Mhz mais le divisor est 20 000 000.
En une seconde, on a 100 000 000 signaux d'horloge, donc en cas d'absence de Reset, c parcourt en 1 seconde 100 000 000/ 20 000 000 = 5 fois les valeurs de divisor (5 cycles). A chaque cycle on a un front montant seulement. On a donc 5 signaux d'horloge par seconde : la fréquence du signal de sortie est donc de 5 Hz.

Q4.

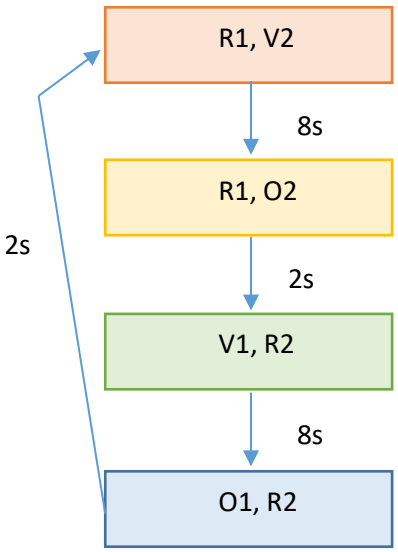


Exercice 2 - Feux de circulation

Q1.

On note R1, O1, V1 les trois états possibles du feux de l'axe A: Rouge, vert et orange.
De même on note R2, V2, O2 les états du feux de l'axe B. A et B sont perpendiculaires.

Les 4 états du système sont les suivants :



C'est le diagramme d'états si on ne prend pas en compte le reset pour le feu de l'axe A et l'axe B.

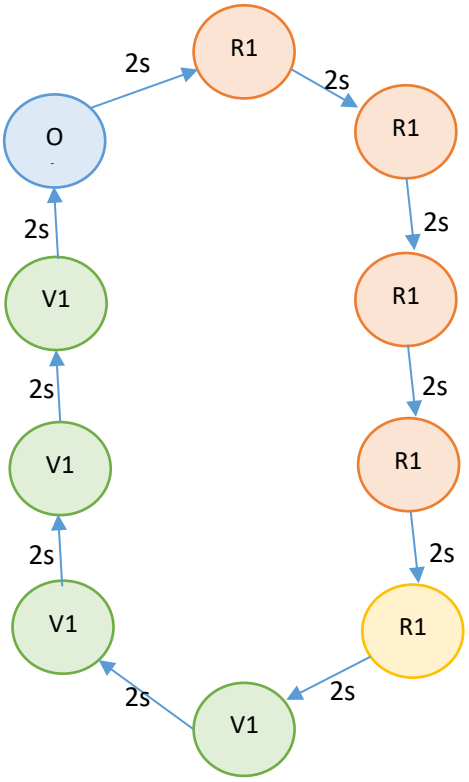
La durée totale d'un cycle (passage d'un feu de rouge à vert, jusqu'au retour à rouge) dure donc 8+2+8+2=20 secondes.

Q2.

La phase la plus courte (qui correspond à lorsqu'un des feux est orange) dure 2 secondes.

La fréquence minimale de l'horloge devrait donc être de deux secondes.

En prenant compte de la fréquence de l'horloge, on obtient le schéma suivant :



Q3.

Explication du code: tout d'abord, on force l'horloge avec le diviseur (fréquence: 100 MHz). Ensuite, pour résoudre le problème de feu de circulation, nous prenons une variable c comme un compteur de temps. À l'état initial, on incrémente c jusqu'à c = 8 pour changer à état 1; même fonctionnement pour les autres états.

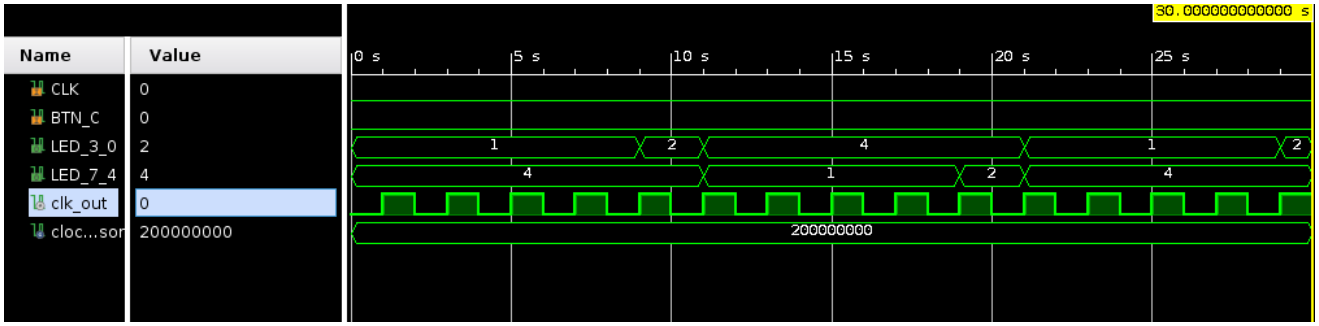
```
ENTITY feu IS
  PORT (
    CLK, BTN_C : IN BIT;
    LED_3_0, LED_7_4 : OUT INTEGER RANGE 0 TO 15
  );
END feu;
ARCHITECTURE Behavioural OF feu IS
  ALIAS reset IS BTN_C;
  SIGNAL clk_out : BIT := '0';
  CONSTANT clock_divisor : INTEGER := 200000000;
BEGIN
  clock_divider : PROCESS (CLK, reset)
  VARIABLE c : INTEGER RANGE 0 TO clock_divisor - 1 := 0;
  BEGIN
    IF reset = '1' THEN
      c := 0;
      clk_out <= '0';
    ELSIF CLK'EVENT AND CLK = '1' THEN
      IF c < (clock_divisor - 1) / 2 THEN
        c := c + 1;
        clk_out <= '0';
      ELSIF c = (clock_divisor - 1) THEN
        c := 0;
        clk_out <= '0';
          ELSE
            c := c + 1;
        clk_out <= '1';
          END IF;
    END IF;
  END PROCESS;
```

```
PROCESS (clk_out)
  VARIABLE c : INTEGER RANGE 0 TO 20; --temps total d'un cycle
  VARIABLE etat : INTEGER RANGE 0 TO 3;
  VARIABLE feux_A, feux_B : INTEGER RANGE 0 TO 4;
   BEGIN
     IF (clk_out'EVENT AND clk_out = '1') THEN
           IF c < 8 THEN
             c := c + 2;
        etat := 0;
       ELSIF c < 10 THEN
         c := c + 2;
         etat := 1;
       ELSIF c < 18 THEN
         c := c + 2;
         etat := 2;
       ELSIF c < 20 THEN
         c := c + 2;
         etat := 3;
       END IF;
       IF c = 20 THEN c := 0;
       END IF;
     END IF;
     CASE etat IS
       WHEN 0 => feux_A := 1; feux_B := 4; --le feu de l'axe A est vert et le feu de l'axe B est rouge
       WHEN 1 => feux_A := 2; feux_B := 4; --le feu de l'axe A est orange, le feu de l'axe B est rouge
       WHEN 2 => feux_A := 4; feux_B := 1; --le feu de l'axe A est rouge, le feu de l'axe B est vert
       WHEN 3 => feux_A := 4; feux_B := 2; --le feu de l'axe A est rouge et le feu de l'axe B est orange
       WHEN OTHERS => feux_A := 1; feux_B := 4;
     END CASE;
     LED_3_0 <= feux_A;
     LED_7_4 <= feux_B;
   END PROCESS;
END Behavioural;
```

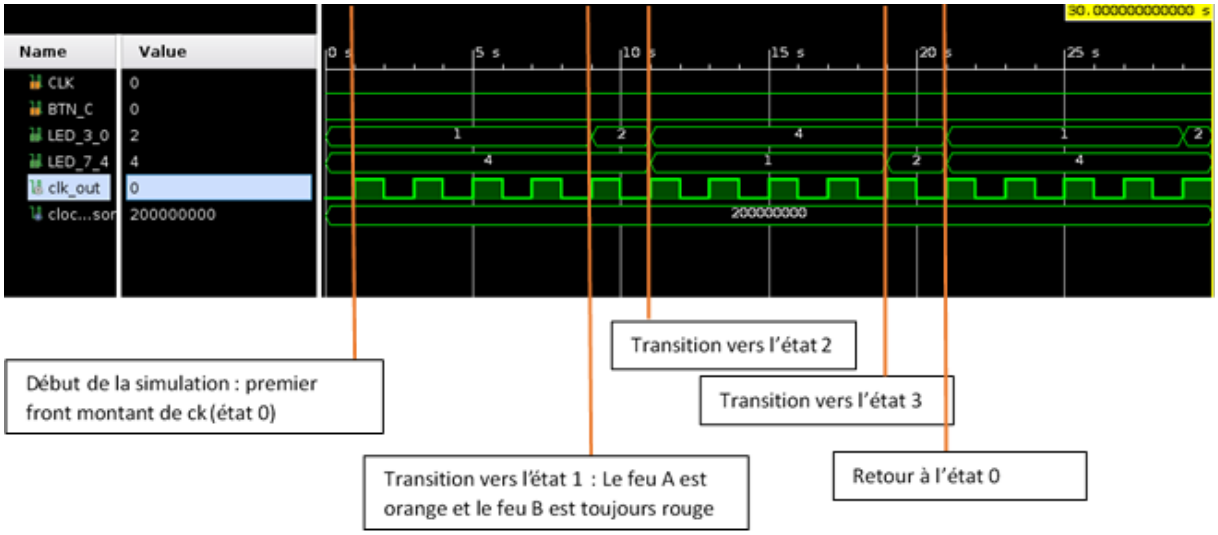
MAINGUET Alice
PHAM Nu-Huyen-Trang MI01 UTC 22/10/19

Q4.

Simulation sans forcer l'horloge de feu, mais en forçant l'horloge de sortie du diviseur:



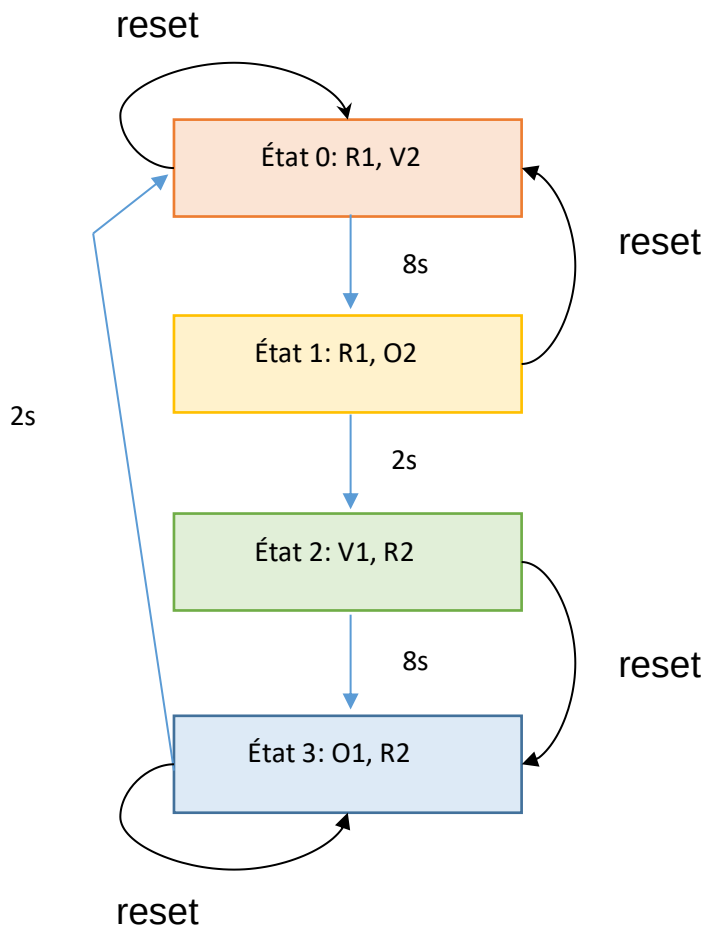
Commentaires :



Q5.

Q6.

On redessine tout d'abord le diagramme d'états avec le reset



Explication: Le reset a pour but de retourner vers l'état initial (R1, V2) mais pour la sécurité des automobilistes, nous avons décidé de changer les états comme dans la figure ci dessus.

À état 0, si reset, il est toujours le même. (cas 1)

À état 1, si reset, on va passer à l'état 0 car l'axe A est rouge: il reste rouge donc les automobilistes continuent à attendre; à l'axe B, le changement de feu orange à feu vert n'est pas dangereux. (cas 2)

À état 2, si reset, on va passer à l'état 3 parce que le changement direct de feu rouge en feu vert et vice versa est dangereux, les gens n'ont pas de temps de réaction pour freiner leur véhicule. C'est pourquoi on passe à l'état 3 pour donner aux gens à l'axe A le feu orange, comme un temps d'attente avant de revenir à l'état 0. (cas 3)

À l'état 3, si reset, on va rester dans même état parce qu'après 2 secondes, il va passer à l'état 0 normalement. Si on passe immédiatement à l'état 0, même explication que l'état 2, c'est dangereux pour les automobilistes. (cas 4)

Code VHDL:

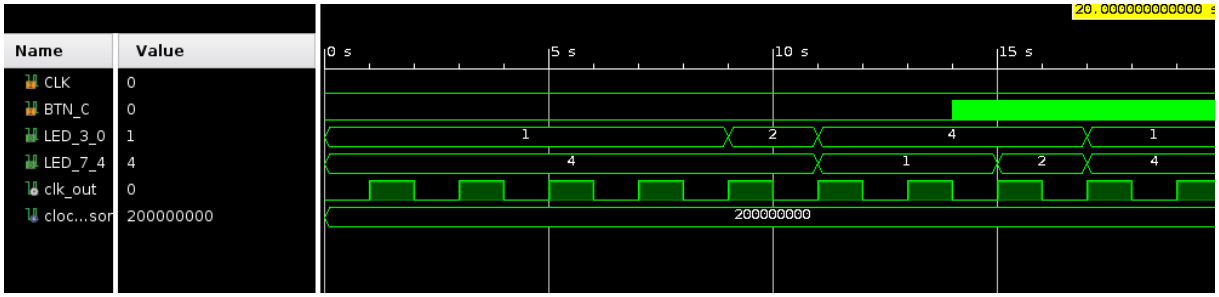
La première partie est même que la question 3

Nous ajoutons seulement la partie différente qui est mise avant la condition d'horloge (reset asynchrone)

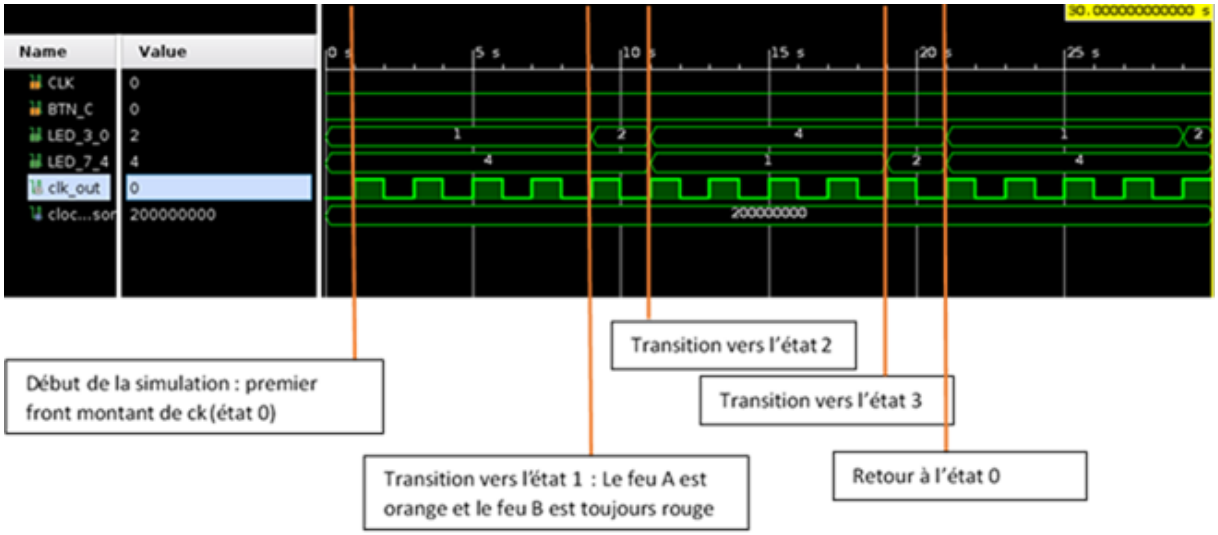
```
IF (reset = '1') THEN
   IF etat = 2 THEN
        etat := 3;
        c := 18;
   ELSIF (etat = 0 OR etat = 1) THEN
        c := 0;
        etat := 0;
   ELSE
        c := c - 1;
        etat := 4;
   END IF;
END IF;
```

Q7.

Simulation avec reset:



avec commentaires:

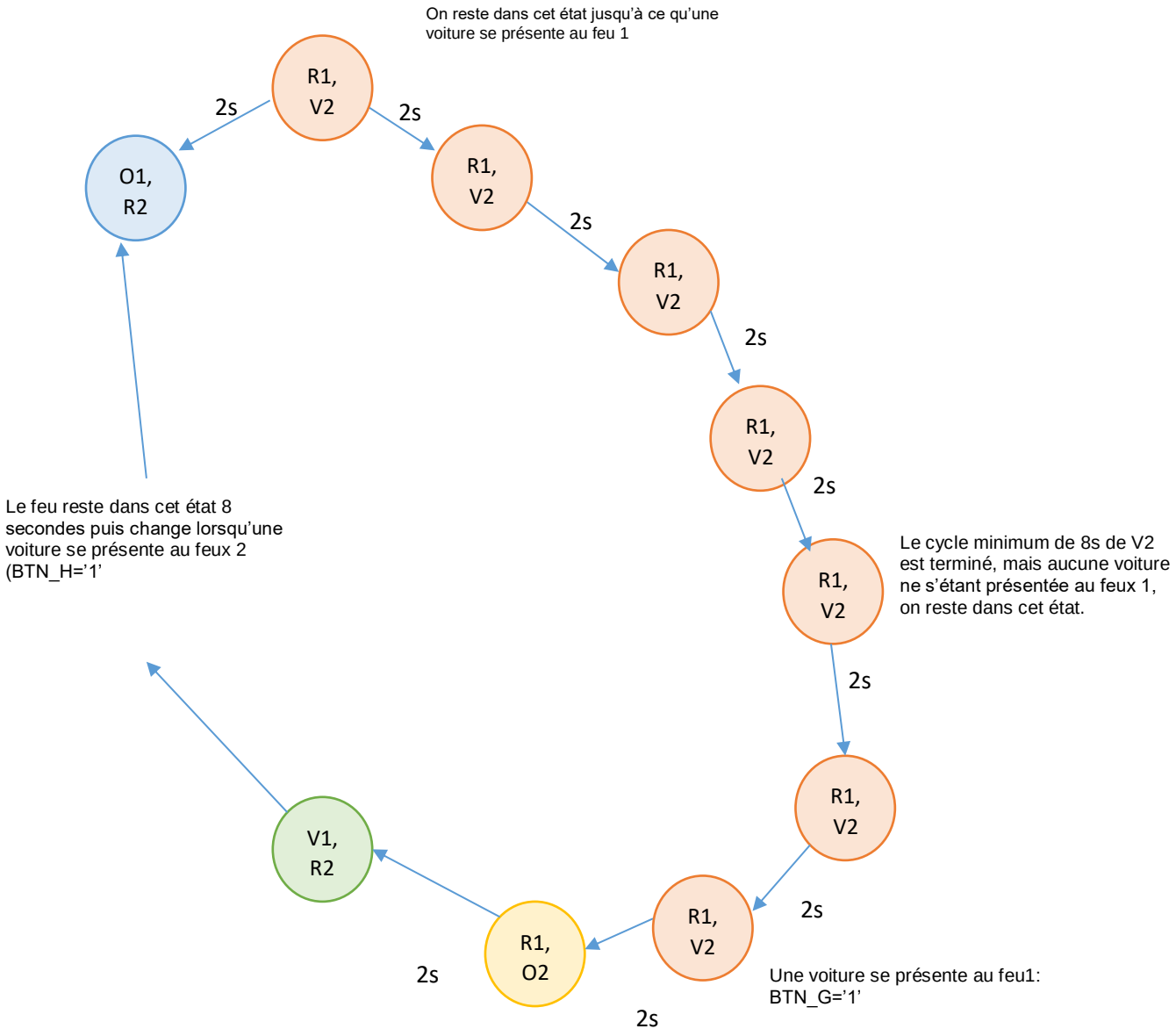


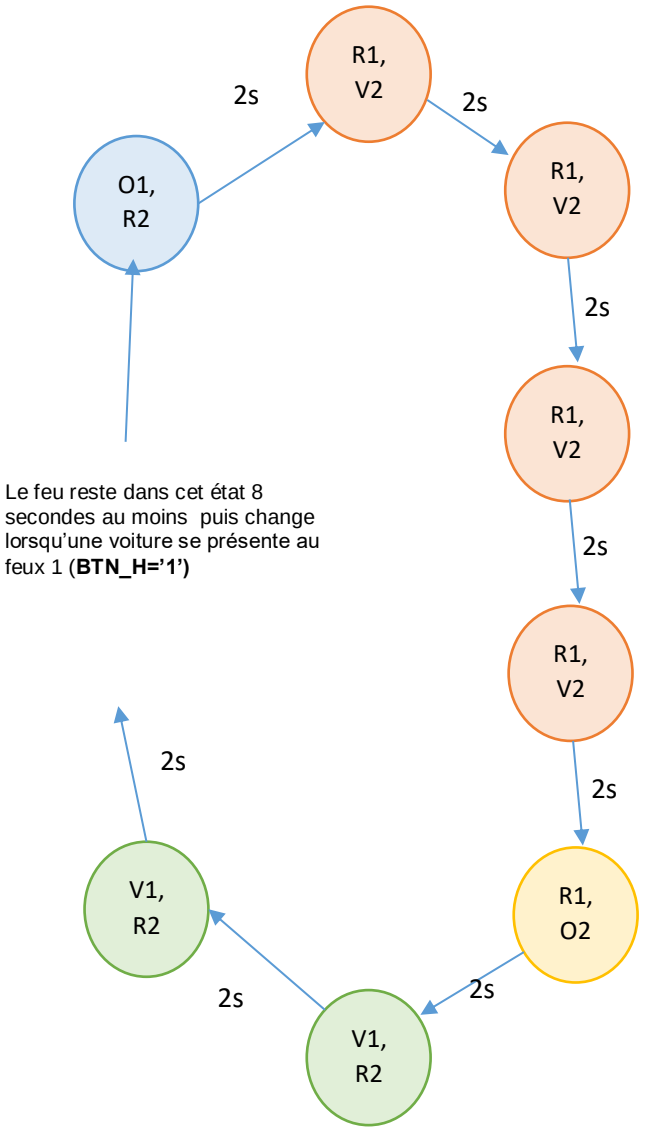
MAINGUET Alice
PHAM Nu-Huyen-Trang MI01 UTC 22/10/19

Exercice 3 - Prise en compte d'un capteur de voiture

Q1.

Nous ne prenons pas en compte le reset maintenant. Les feux de circulation sont maintenant équipés d'un capteur de voiture. Quand il n'y a pas voitures à l'axe A par exemple, on aura toujours le feu vert de l'axe B jusqu'à ce qu'on détecte le signal de véhicules sur l'axe A. Même chose pour les véhicules de l'axe B. Cependant en cas de changement de couleur de feu lorsqu'une voiture arrive, le feu vert doit passer au orange, et respecter les 2 secondes de orange avant de passer au rouge. De plus, les feux ne peuvent pas changer de couleur en cas de présence d'une voiture si la durée minimale du vert (8s) n'a pas été respectée.
Voici deux schémas représentant des situations possibles:





Q2.

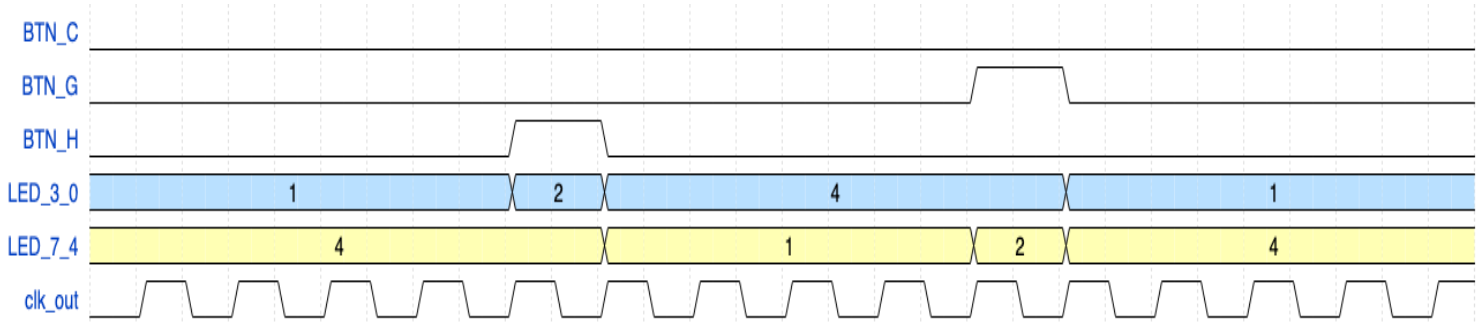
Le code est presque même que l'exercice précédente, on juste ajoute la condition de BTN_H ou BTN_D.

Nous ajoutons seulement la partie différente que l'exercice précédente.

```
IF (clk_out'EVENT AND clk_out = '1') THEN
        IF c < 8 THEN
          c := c + 2;
          etat := 0;
        ELSIF c < 10 AND BTN_H = '1' THEN
          c := c + 2;
          etat := 1;
        ELSIF c < 18 THEN
          c := c + 2;
          etat:=2;
        ELSIF c < 20 AND BTN_G = '1' THEN
          c := c + 2;
          etat := 3;
        END IF;
        IF c = 20 THEN
           c := 0;
        END IF;
END IF;
```

Q3.

Simulation sans reset



Commentaires: En réalité, pour BTN_H, BTN_G, il reste aussi le temps d'attente de traitement de ces signaux avant de changer d'état. Dans cette simulation à la main, nous avons supposé que nous faisons ce diagramme dans l'unité second (s) avec la période 2s pour l'horloge, c'est pourquoi on ne peut pas voir clairement le temps d'attente (il est trop petit)

Q4.

FPGA + Schéma RTL

Q5.

Selon nous, le reset asynchrone est plus efficace. Premièrement, comme nous avons expliqué dans l'exercice précédente, nous pensons à la sécurité des automobilistes. Si le reset est synchrone, on doit attendre le front d'horloge pour le prendre en compte, cela fait perdre du temps des gens. Nous voulons que notre système de feux de circulation ne prend pas beaucoup de temps des gens mais la sécurité est prioritaire.

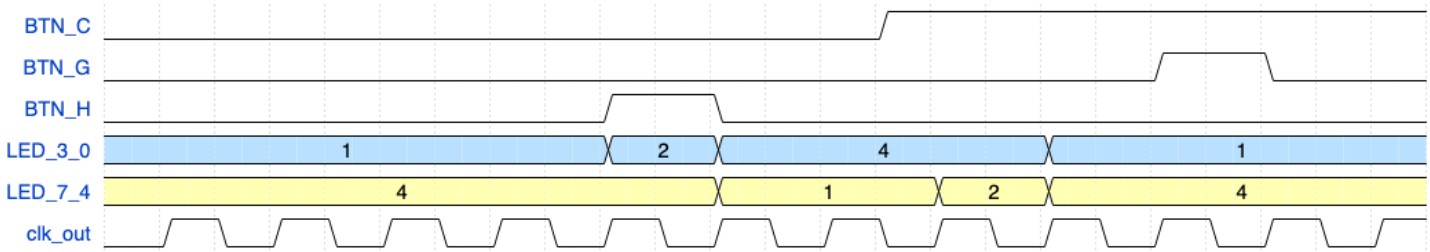
Q6.

Le code est le même que la question 6 de l'exercice 2, nous devons ajouter seulement la partie ci-dessous après la condition de reset

```
IF (clk_out'EVENT AND clk_out = '1') THEN
        IF c < 8 THEN
          c := c + 1;
          etat := 0;
        ELSIF c < 10 AND BTN_H = '1' THEN
          c := c + 1;
          etat := 1;
        ELSIF c < 18 THEN
          c := c + 1;
          etat := 2;
        ELSIF c < 20 AND BTN_G = '1' THEN
          c := c + 1;
          etat := 3;
        END IF;
        IF c = 20 THEN
           c := 0;
        END IF;
END IF;
```

MAINGUET Alice
PHAM Nu-Huyen-Trang MI01 UTC 22/10/19

Simulation:



Dans cette simulation, on peut voir que quand le reset est activé, on attend une seconde pour la raison de sécurité pour les automobilistes, puis on va remettre à l'état 2 avant de l'état 3 (cas 3), après 2 secondes de feu orange, on va retourner à l'état initial. À l'état 0, même si BTN_G est activé, on ne le prend pas en compte car il faut que BTN_H soit activé pour changer l'état.