Nom : Prénom : Matricule :

INF3500 : Conception et réalisation de systèmes numériques

Examen intra #2 – 17 mars 2022

Q1	
Q2	
Q3	
Q4	
Total	

Durée: 1 heure. Documentation: Une feuille recto verso 8.5"×11" ou A4 permise.
Pondération: 10%. Calculatrice: Programmable permise.
Directives particulières:

a. Ordinateurs interdits. Appareils mobiles interdits.
b. Répondre à toutes les questions, la valeur de chaque question est indiquée.
c. Répondre sur le questionnaire et le remettre.
d. Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement vos suppositions.

Question 1. (2 points – 12 minutes)

Un système numérique est utilisé pour surveiller le trafic routier sur un pont et identifier les véhicules plus à risque de provoquer un accident. Pour chaque véhicule détecté, le système a 5 entrées: (1) le type de véhicule : {camion, voiture, moto, vélo}, (2) la vitesse du véhicule arrondie au km/h près {0 à 511}, (3) l'heure de la journée arrondie à l'heure prêt {0 à 23}, (4) le jour de la semaine {0 à 6}, et (5) le jour de l'année {1 à 365, où 1 est le 1er janvier} – pour les besoins de cette question, on ne tient pas compte des années bissextiles.

La sortie du système est binaire : risque faible (0) ou risque élevé (1).

Les règles sont les suivantes:

- Le jour (6 h à 18 h), la vitesse maximale sur le pont est de 70 km/h pour les camions et 90 km/h pour les voitures et motos.
- Le soir et la nuit (19 h à 5 h), la vitesse maximale sur le pont est de 60 km/h pour les camions et 80 km/h pour les voitures et motos.
- Le dimanche matin, de 9 h à midi, seuls les vélos sont permis sur le pont.
- Les camions sont interdits pendant les heures de pointe, de 6 h à 9 h et de 16 h à 19 h.
- Les motos sont interdites du 1er décembre au 31 mars, (jours 335 à 365 et 1 à 90).

Proposez un partitionnement en classes pour chacune des 5 entrées. Minimisez le nombre de classes.

Type de véhicule : _____

Vitesse du véhicule : _____

Heure de la journée : _____

Jour de la semaine : _____

Jour de l'année : _____

Combien de vecteurs de tests sont nécessaires pour effectuer un test fort ? Justifiez votre réponse.

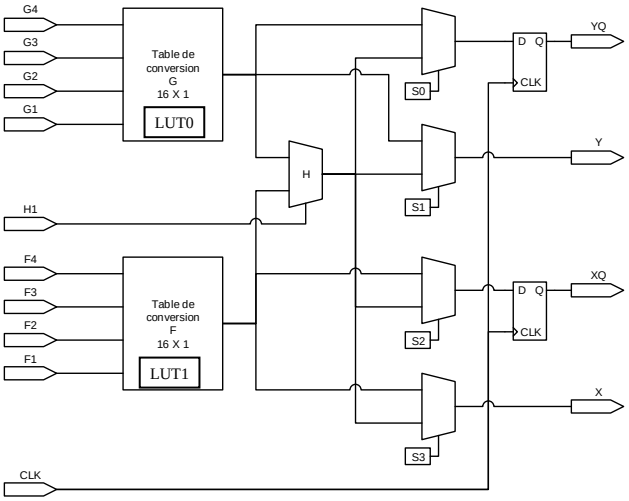
Question 2. (3 points – 18 minutes)

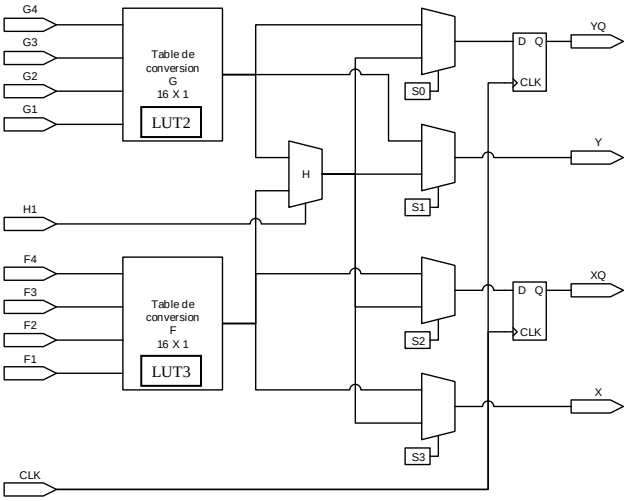
Annotez le diagramme suivant pour montrer comment implémenter le code VHDL. Indiquez clairement les entrées, les sorties, et le cheminement des signaux à l'intérieur et entre les tranches du modèle de FPGA. Il est possible d'effectuer des connexions de la sortie d'une tranche (par exemple, d'une sortie Y) vers l'entrée d'une table de conversion (*Look-Up Table* – LUT) d'une autre tranche (par exemple, une entrée F).

Montrez, dans le tableau à droite du code, le contenu de chaque LUT que vous utilisez.

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

entity cctsequentielex6 is
  port (
    clk, reset : in STD_LOGIC;
    s, t : in std_logic;
    op : in unsigned(2 downto 0);
    A, B, C : out std_logic
  );
end cctsequentielex6;

architecture arch of cctsequentielex6 is
begin
  process(CLK, reset) is
  begin
    if (reset = '1') then
      A <= '0';
      C <= '0';
    elsif (rising_edge(CLK)) then
      case to_integer(op(1 downto 0)) is
        when 0 => A <= s or t;
        when 1 => A <= s and t;
        when 2 => A <= s xor t;
        when others => A <= not(s);
      end case;
      C <= (s xor t)
          or (op(2) and op(1) and op(0));
    end if;
  end process;

  with op(2) select
  B <= s when '1', t when others;

end arch;
```

G4/F4	G3/F3	G2/F2	G1/F1	LUT3	LUT2	LUT1	LUT0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				





Question 3. (3 points : 18 minutes)

Le processeur PolyCrypto prend en entrée deux mots `A0` et `B0` de 32 bits et leur applique 15 rondes de manipulations logiques pour les chiffrer à l'aide d'une clé secrète. Une version en pseudocode de l'algorithme PolyCrypto est donnée ici.

La clé est emmagasinée dans une mémoire ROM de 16 mots de 32 bits, qui peut être vue comme une table de vérité à 4 entrées (les 4 bits d'adresse) et 32 sorties.

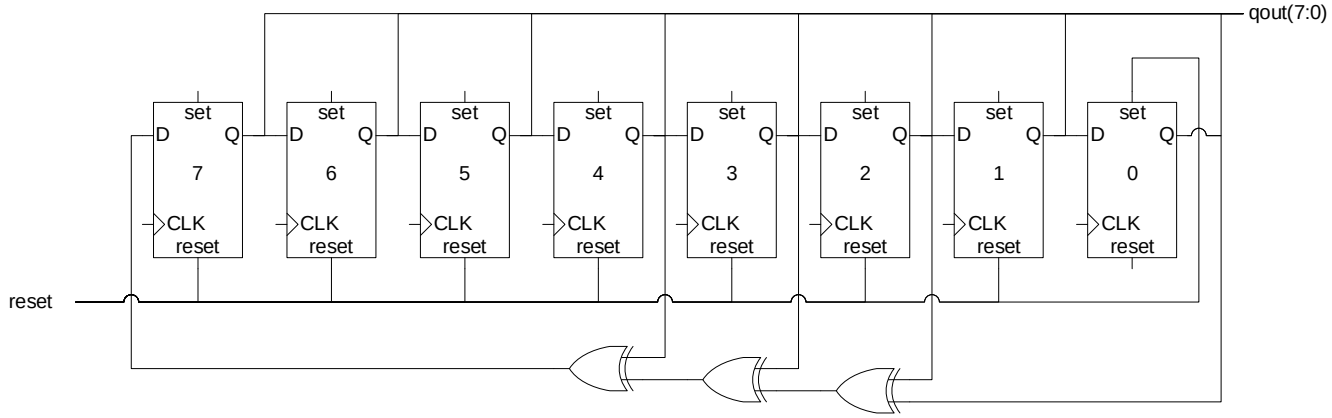
Les calculs doivent débuter quand le signal d'entrée `init` est activé. Le signal de sortie `fini` doit être activé quand les calculs sont terminés et que les registres `P` et `Q` contiennent le résultat final.

Faites la conception d'un chemin des données correspondant à cet algorithme et donnez son diagramme.

```
si init == '1' {
    A ← A0;
    B ← B0;
    K ← 0;
    fini ← '0';
} sinon, à chaque coup d'horloge {
    si K < 15 {
        B ← A;
        A ← A xor ROM(K) xor B;
        K ← K + 1;
    } sinon {
        fini ← '1';
        P ← A;
        Q ← B;
    }
}
```

Question 4. (2 points – 12 minutes)

Considérez le circuit d'un registre à décalage à rétroaction linéaire (*Linear Feedback Shift Register* – LFSR) :



Donnez le code VHDL synthétisable d'une architecture correspondant à la déclaration d'entité suivante et au diagramme. L'action du signal reset doit être asynchrone. L'action du signal reset doit être asynchrone et activée sur une '1' (actif haut). Les bascules sont sensibles au front montant d'horloge.

```
library ieee;
use ieee.std_logic_1164.all;

entity lfsr_8_bits is
 port(
 clk, rst : in  std_logic;
 qout : out std_logic_vector(7 downto 0)
 );
end lfsr_8_bits;
```

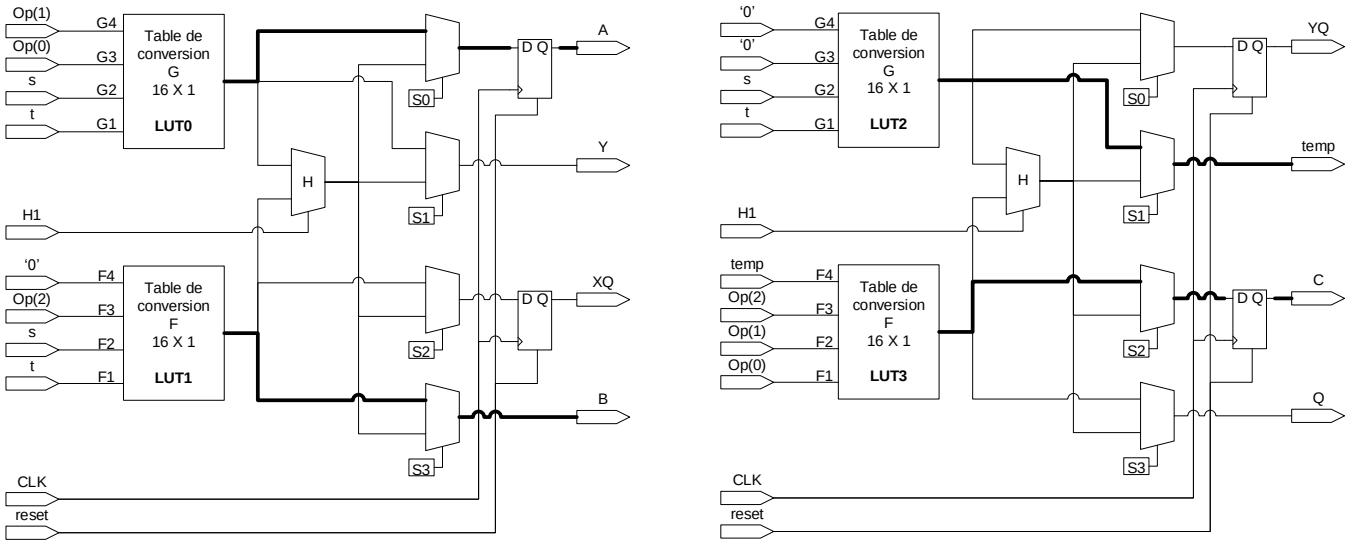
Solutions

#1 Partitionnement en classes

(1) le type de véhicule : {camion}, {voiture}, {moto}, {vélo}
(2) la vitesse du véhicule {0 à 60}, {61 à 70}, {71 à 80}, {81 à 90}, {91 à 511}
(3) l'heure de la journée arrondie à l'heure prêt {0 à 5}, {6 à 8}, {9 à 11}, {12 à 15}, {16 à 18}, {19 à 23}
On pourrait groupe les classes 0-5 et 19-23 en une seule classe.
(4) le jour de la semaine {0 à 5}, {6}, ou {0 à 4}, {5}, {6}
(5) le jour de l'année {1 à 90}, {91 à 334}, {335 à 365};
On pourrait grouper les classes 1-90 et 91-334 en une seule classe.

#2 Code VHDL séquentiel vers FPGA

Une solution possible



G4/F4	G3/F3	G2/F2	G1/F1	LUT3	LUT2	LUT1	LUT0
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	0	1
0	0	1	1	0	0	1	1
0	1	0	0	0	-	0	0
0	1	0	1	0	-	0	0
0	1	1	0	0	-	1	0
0	1	1	1	1	-	1	1
1	0	0	0	1	-	-	0
1	0	0	1	1	-	-	1
1	0	1	0	1	-	-	1
1	0	1	1	1	-	-	0
1	1	0	0	1	-	-	1
1	1	0	1	1	-	-	1
1	1	1	0	1	-	-	0
1	1	1	1	1	-	-	0

#3 PolyCrypto

Il y a plusieurs solutions possibles.

Ici, pour les registres A, B et K, on utilise une combinaison d'un multiplexeur à l'entrée avec un signal de chargement. Pour les registres P, Q et fini on utilise seulement un multiplexeur à l'entrée.

#4 LFSR

Il y a plusieurs réponses possibles.

```
library ieee;
use ieee.std_logic_1164.all;

entity lfsr_8_bits is
 port(
 clk, rst : in  std_logic;
 qout : out std_logic_vector(7 downto 0)
 );
end lfsr_8_bits;

architecture arch of lfsr_8_bits is
begin
 process(clk, rst)
 variable reg : std_logic_vector(7 downto 0);
 begin
    if rst = '1' then
       reg := "00000001";
    elsif rising_edge(clk) then
    reg := (reg(0) xor reg(2) xor reg(3) xor reg(4)) & reg(7 downto 1);
    end if;
    qout <= reg;
 end process;
end arch;

architecture arch_sig of lfsr_8_bits is
signal reg : std_logic_vector(7 downto 0);
begin
   process(clk, rst)
   begin
       if rst = '1' then
          reg <= "00000001";
       elsif rising_edge(clk) then
          reg <= (reg(0) xor reg(2) xor reg(3) xor reg(4)) & reg(7 downto 1);
       end if;
   end process;
   qout <= reg;
end arch_sig;
```