Nom : _____ Matricule : _____

INF3500 : Conception et réalisation de systèmes numériques

Examen intra - vendredi 21 octobre 2011

Durée: 2h; Pondération: 20%

Documentation: Une feuille recto verso 8.5"×11" ou A4 permise.

Ordinateurs interdits. Calculatrice: Programmable permise.

Répondre à toutes les questions, la valeur de chaque question est indiquée.

Répondre sur le questionnaire et le remettre. Vous pouvez utiliser le verso du questionnaire si nécessaire. Ne détachez pas de pages du questionnaire.

Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.

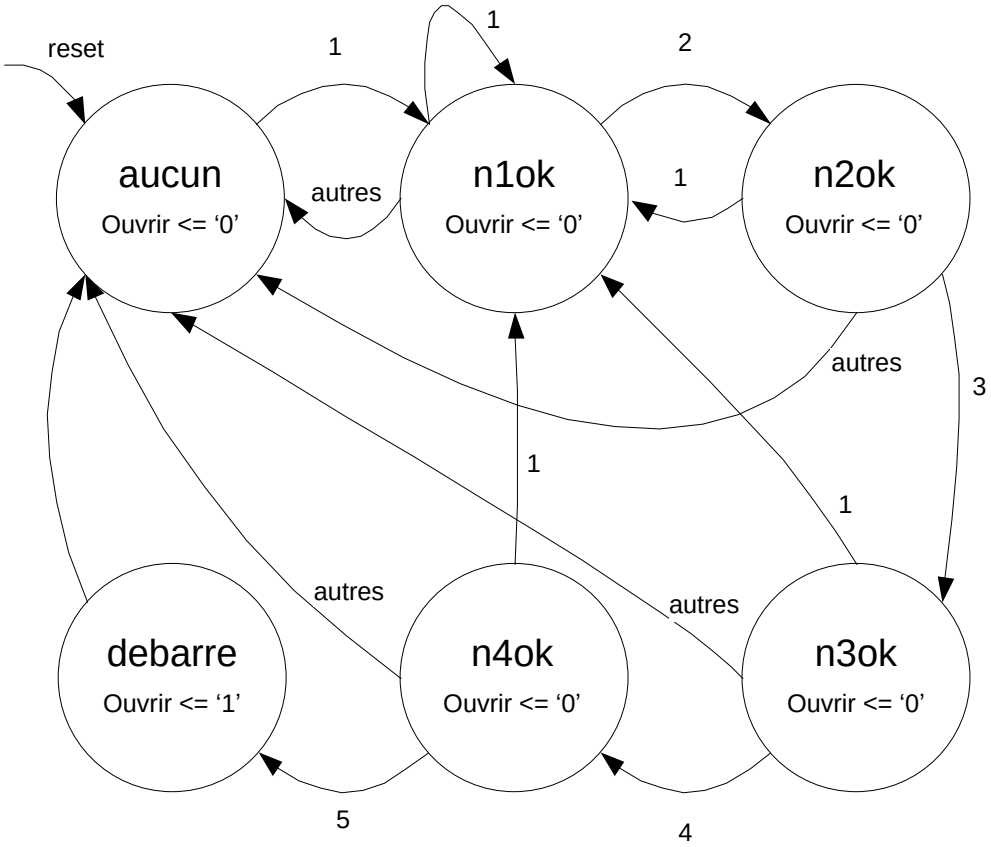
Question 1. (4 points)

Considérez le problème de la conception d'un cadenas numérique à cinq numéros entre 0 et 9, dont la combinaison est 1-2-3-4-5. Le cadenas est doté d'un signal de réinitialisation `reset`, d'un bouton `entrer` qui fait office d'horloge, et d'un port `numero` de quatre bits permettant de spécifier le numéro. Une sortie `ouvrir` indique si le cadenas doit être déverrouillé ou non.

Pour débarrer le cadenas, un utilisateur devrait placer le nombre 1 sur le port `numero`, appuyer sur `entrer`, placer le nombre 2 sur le port `numero`, appuyer sur `entree`, et ainsi de suite jusqu'à placer le nombre 5 sur le port `numero` et appuyer sur `entrer`. Le signal `ouvrir` passerait alors de 0 à 1, et resterait à 1 tant que l'utilisateur n'appuierait pas sur `entrer` à nouveau.

a. Donnez un diagramme d'états pour le cadenas. Vous devez utiliser une machine de Moore.

Solution :



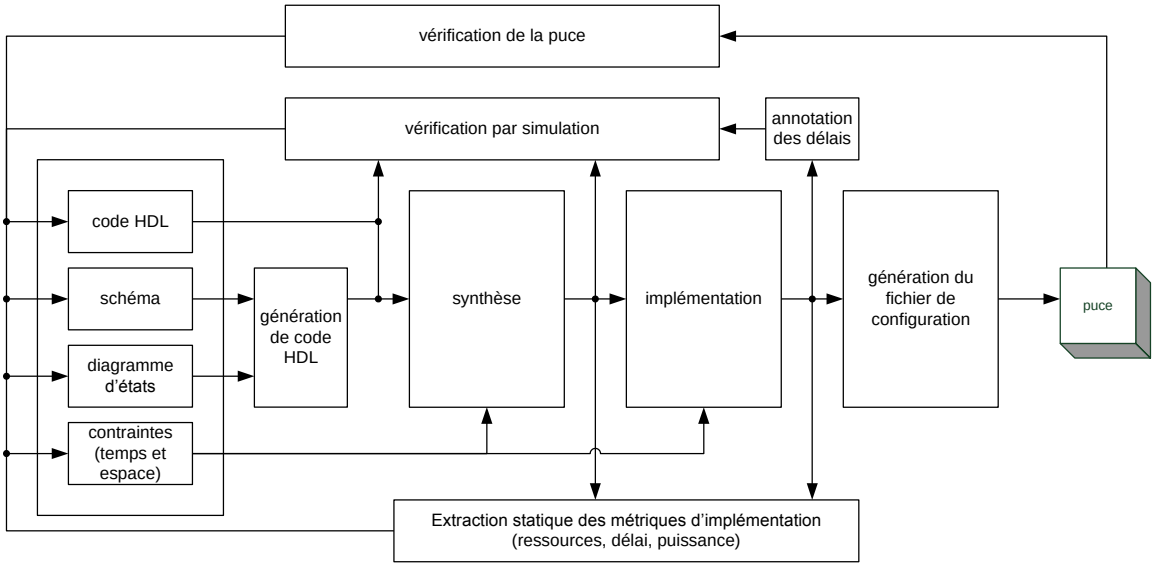
b. Donnez une architecture synthétisable pour l'entité VHDL suivante correspondant à la spécification. Des points seront accordés pour une architecture qui permet de changer facilement la combinaison.

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

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

entity cadenas is
    port (
        reset, entrer : in STD_LOGIC;
        numero : in unsigned(3 downto 0);
        ouvrir : out STD_LOGIC
    );
end cadenas;

architecture arch of cadenas is
type combinaison is array (natural range <>) of natural range 0 to 15;
constant lacombine: combinaison := (1, 2, 3, 4, 5);
type type_etat is (aucun, n1ok, n2ok, n3ok, n4ok, debarre);
signal etat : type_etat := aucun;
begin
    ouvrir <= '1' when etat = debarre else '0';
    process(entrer, reset) is
    begin
        if (reset = '0') then
           etat <= aucun;
        elsif (rising_edge(entrer)) then
           case etat is
              when aucun =>
                 if numero = lacombine(0) then etat <= n1ok;
                 else etat <= aucun;
                 end if;
              when n1ok =>
                 if numero = lacombine(1) then etat <= n2ok;
                 elsif numero = lacombine(0) then etat <= n1ok; -- pas dans la spéc.
                 else etat <= aucun;
                 end if;
              when n2ok =>
                 if numero = lacombine(2) then etat <= n3ok;
                 elsif numero = lacombine(0) then etat <= n1ok; -- pas dans la spéc.
                 else etat <= aucun;
                 end if;
              when n3ok =>
                 if numero = lacombine(3) then etat <= n4ok;
                 elsif numero = lacombine(0) then etat <= n1ok; -- pas dans la spéc.
                 else etat <= aucun;
                 end if;
              when n4ok =>
                 if numero = lacombine(4) then etat <= debarre;
                 elsif numero = lacombine(0) then etat <= n1ok; -- pas dans la spéc.
                 else etat <= aucun;
                 end if;
              when debarre =>
                 etat <= aucun;
              when others =>
                 etat <= aucun;
           end case;
        end if;
    end process;
end arch;
```

Question 2. (3 points)

a. Illustrez, avec un diagramme, le flot de conception que vous avez suivi dans les laboratoires du cours INF3500 à ce jour, de la description du design à la vérification du fonctionnement de la puce.

Solution :



b. Dans le flot de conception, énumérez les différentes entrées possibles à l'étape de vérification par simulation et expliquez leurs différences.

Solution :

1. Après la création/génération de code HDL. Vérification fonctionnelle seulement.

2. Après la synthèse. Vérification fonctionnelle des blocs synthétisés. Permet de vérifier que la synthèse a bien fonctionné et que le produit de la synthèse est conforme aux intentions du concepteur.

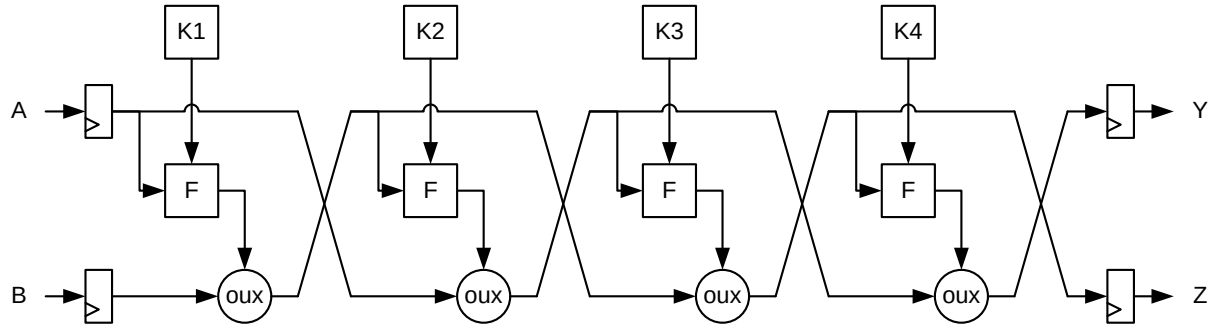
3. Après l'implémentation. Vérification fonctionnelle avec annotation des délais estimés suite au placement et au routage des blocs synthétisés. Les chronogrammes permettent de voir comment la propagation des délais se fait dans le circuit. Cette vérification permet aussi de vérifier que l'implémentation a bien fonctionné, et que son produit est conforme aux intentions du concepteur.

c. Selon votre expérience dans les laboratoires d'INF3500, énumérez trois métriques d'implémentation en termes de ressources matérielles utilisées.

Réponse :
#LUTs, #bascules, #tranches (*slices*), #multiplieurs, #blocs d'entrée/sortie, #mémoire BRAM

Question 3. (4 points)

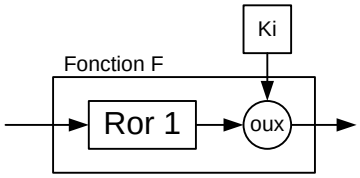
Le réseau de Feistel est utilisé dans les algorithmes de chiffrement par bloc. Plusieurs algorithmes utilisent le réseau de Feistel, dont DES, Blowfish et RC5. Le diagramme suivant illustre un réseau de Feistel simple à quatre étages. Les algorithmes cryptographiques basés sur un réseau de Feistel diffèrent principalement dans le nombre d'étages et dans la nature de la fonction F.



Le message à chiffrer est décomposé en un flux de nombres appliqués aux entrées A et B du réseau à chaque coup d'horloge. Les quatre clés secrètes K1, K2, K3 et K4 restent normalement constantes pour le chiffrement du message. Les sorties Y et Z sont un flux de nombres (une sortie par coup d'horloge) représentant le message chiffré.

On constate qu'à chaque étage le signal du haut est combiné à la clé par la fonction F. On effectue ensuite un ou-exclusif bit à bit avec le signal du bas. À la fin de l'étage, les signaux du haut et du bas sont interchangés pour le prochain étage.

Supposez ici que tous fils du diagramme représentent un signal de 16 bits. Supposez que la fonction F consiste à appliquer une rotation de l'entrée de 1 bit vers la droite puis à effectuer l'opération ou-exclusif bit à bit avec la clé K_i exprimée sur 16 bits.



Considérez la déclaration d'entité suivante pour le diagramme du réseau de Feistel. Donnez une architecture en VHDL synthétisable correspondant à cette entité et au diagramme.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity feistel4 is
    generic (
        W : positive := 16
    );
    port (
        clk, reset : in std_logic;
        A, B : in unsigned(W - 1 downto 0);
        Y, Z : out unsigned(W - 1 downto 0);
        K1, K2, K3, K4: unsigned(W – 1 downto 0)
    );
end feistel4;
```

(répondre à la page suivante)

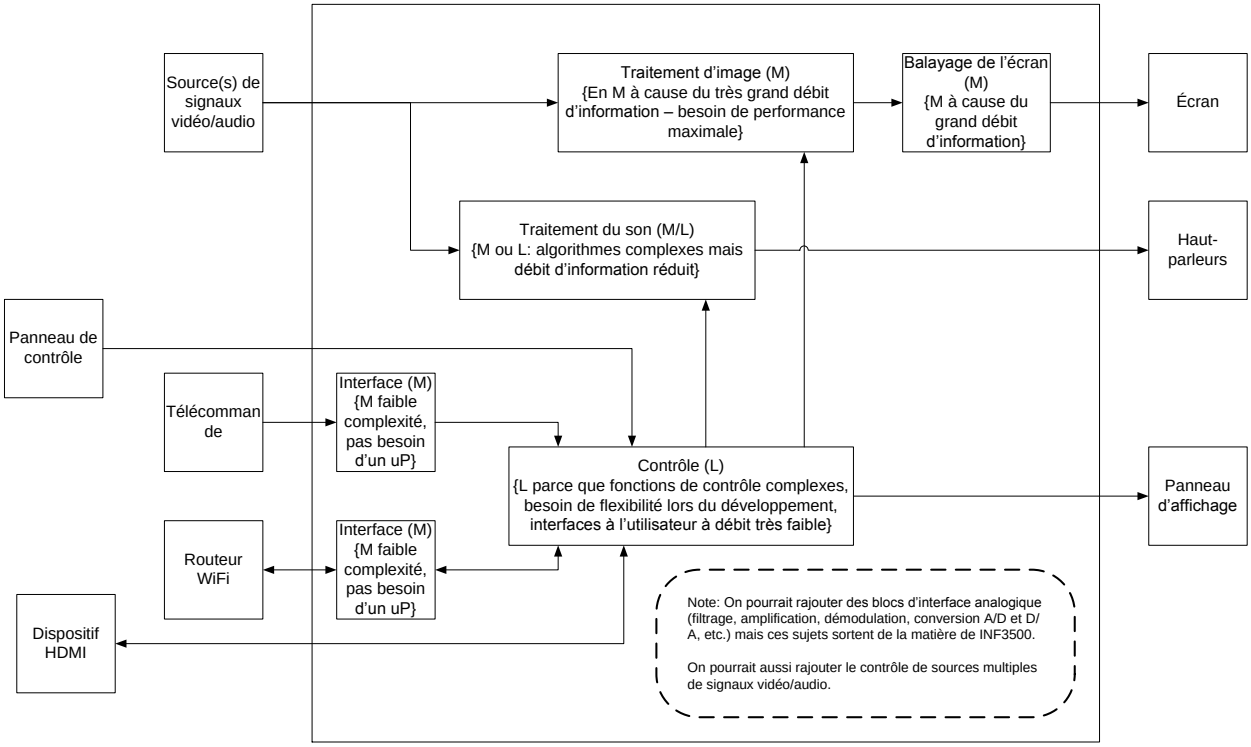
Réponse à la question 3 :

```
arch monarch of feistel4 is

type unsigned2D is array(natural range <>) of unsigned(W - 1 downto 0);

signal A0, A1, A2, A3, A4, B0, B1, B2, B3, B4 : unsigned(W - 1 downto 0);

begin

   -- registres à l'entrée pour A et B et à la sortie pour Y et Z
   process (clk, reset)
   begin
      if reset = '1' then
         Y <= (others => '0');
         Z <= (others => '0');
         A0 <= (others => '0');
         B0 <= (others => '0');
      elsif rising_edge(clk) then
         A0 <= A;
         B0 <= B;
         Y <= A4;
         Z <= B4;
      end if;
   end process;

   -- description du réseau comme tel
   -- on pourrait aussi paramétrer A et B, puis utiliser une description
   -- avec une boucle (dans un process) ou bien avec énoncé generate
   B1 <= A0;
   A1 <= (A0 ror 1) xor K1 xor B0;

   B2 <= A1;
   A2 <= (A1 ror 1) xor K2 xor B1;

   B3 <= A2;
   A3 <= (A2 ror 1) xor K3 xor B2;

   B4 <= A3;
   A4 <= (A3 ror 1) xor K4 xor B3;

end monarch;
```

Question 4. (3 points)

Considérez le problème de la conception du système numérique contrôlant un téléviseur à haute définition. Énumérez les entrées et les sorties. Proposez une décomposition de ce système en modules et faites un partitionnement initial logiciel-matériel. Pour chaque module, justifiez brièvement votre choix entre une implémentation en logiciel (L) et une implémentation en matériel (M).

Solution :



Question 5. (6 points)

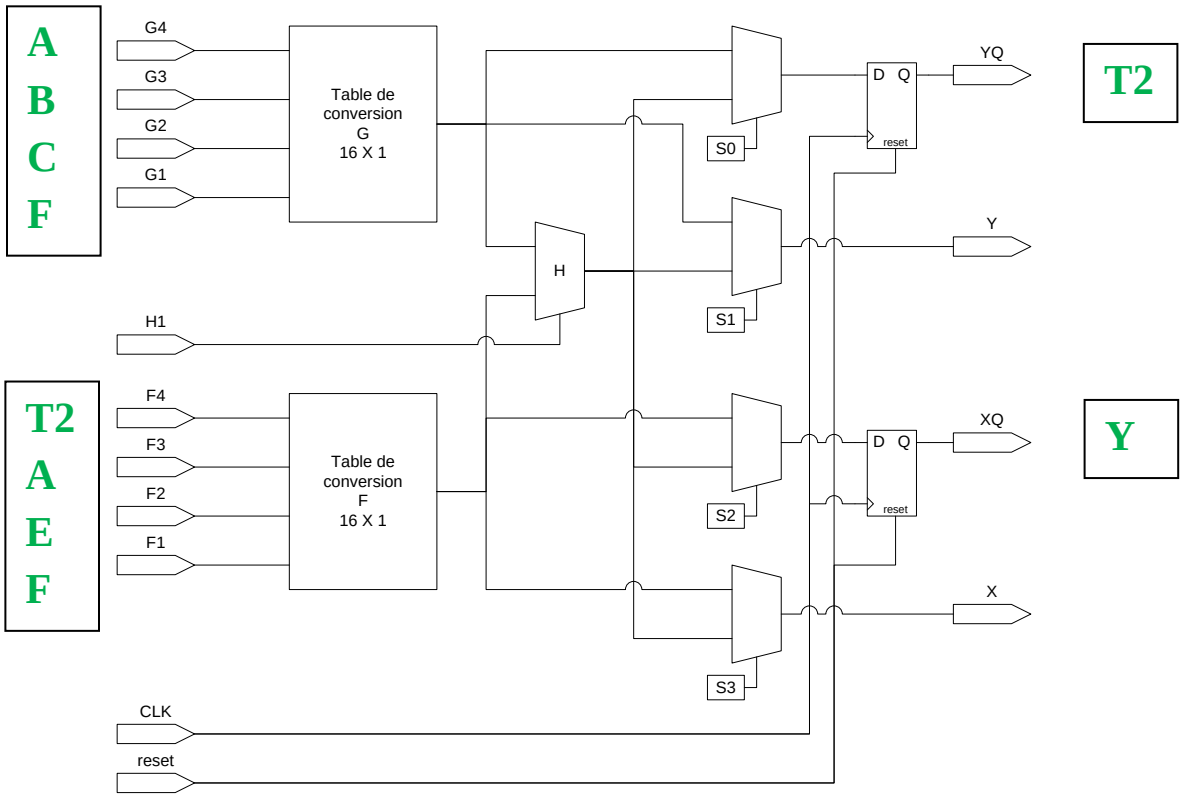
Considérez le code VHDL suivant. Montrez, sur le modèle de FPGA fourni aux pages suivantes, un résultat possible de la synthèse et de l'implémentation de ce code.

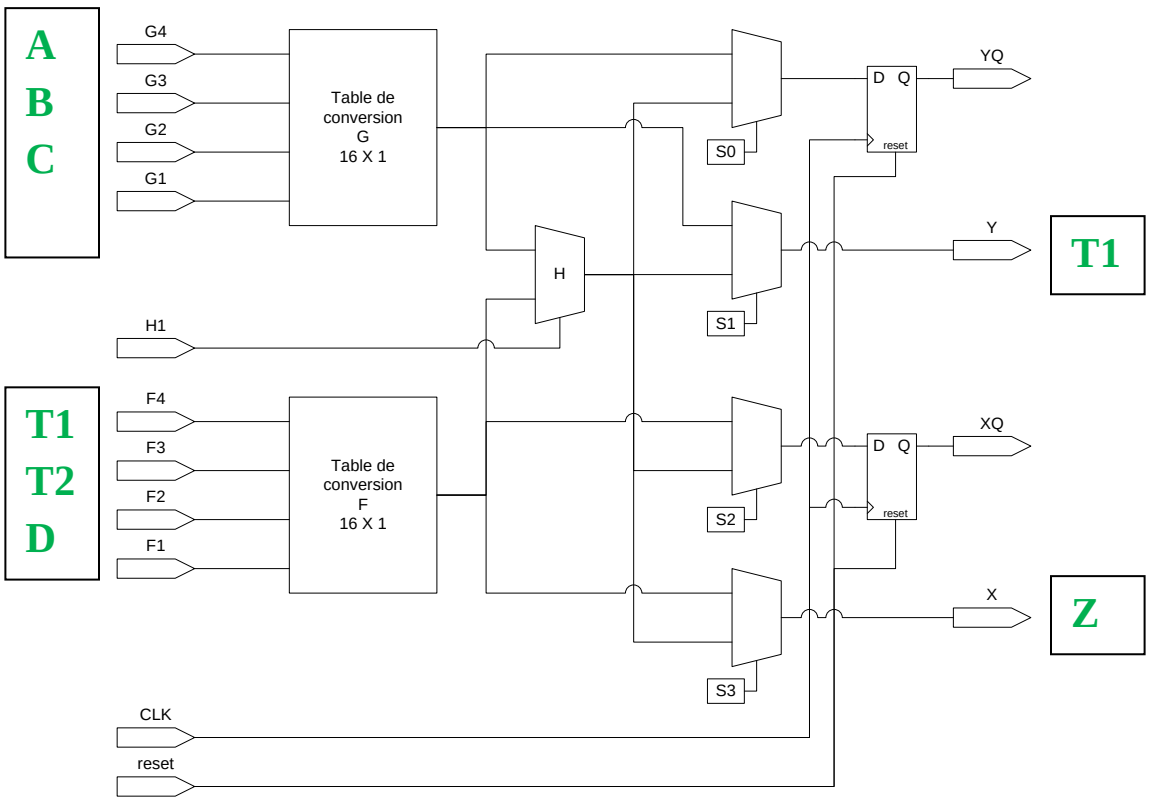
a. Indiquez directement sur le dessin où chaque signal se situe ainsi que les interconnections entre les blocs.

b. Indiquez dans les tables de vérité fournies le contenu de chacune des tables de conversion que vous utilisez.

```
library ieee;
use ieee.std_logic_1164.all;
entity monModule6 is
   port (
      reset, clk: in std_logic;
      A, B, C, D, E, F : in std_logic;
      Y, Z : out std_logic
   );
end monModule6;
architecture arch of monModule6 is
signal T1, T2 : std_logic;
begin
   T1 <= (A and B) or C;
   Z <= (T1 or T2) and D;
   process(CLK, reset) is
   begin
       if (reset = '1') then
          T2 <= '0';
          Y <= '0';
       elsif (rising_edge(CLK)) then
          T2 <= (A and B and C) xor F;
          Y <= T2 and A and E and F;
       end if;
   end process;
end arch;
```

Réponse à la Question 5 a.





Réponse à la question 5 b.

G4 (A)	G3 (B)	G2 (C)	G1 (F)	G (T2)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

F4 (T2)	F3 (A)	F2 (E)	F1 (F)	F(Y)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

G4 (A)	G3 (B)	G2 (C)	G1 ( )	G (T1)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

F4 (T1)	F3 (T2)	F2 (D)	F1 ( )	F (Z)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1