INF3500 : Conception et réalisation de systèmes numériques

Examen final

23 avril 2009

Durée: 2h30.
Pondération: 40%.
Documentation: Une feuille recto verso 8.5”×11” ou A4 manuscrite permise.
Calculatrice: Programmable permise.
Directives particulières: Ordinateurs interdits, répondre à toutes les questions, la valeur de chaque question est indiquée.
Ne posez pas de questions. En cas de doute sur le sens d’une question, énoncez clairement toute supposition que vous faites.

Question 1. (12 points) – Processeur à but spécifique

Dans les applications de traitement de signaux, une opération typique consiste à trouver la somme des produits entre un signal d'entrée et celle d'un tableau de constantes tel que :

$$\text{Re}\,sultat = \sum_{i=0}^{i=N-1} Entree_i \times Coef_i$$

Voici l'entité du processeur à but spécifique que vous devez concevoir.

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

entity sommeproduit is
generic (
        W : positive;
        M : positive := 8);

port (
       reset, CLK : in STD_LOGIC;
       entree : in unsigned(W - 1 downto 0);
       go : in std_logic;
       entreevalide : in std_logic;
       sortievalide : out std_logic;
       erreur : out std_logic;
       sortie : out unsigned(W - 1 downto 0)
);
end sommeproduit;
```

Le paramètre générique ***W*** spécifie le nombre de bits des ports *entree* et *sortie*.

Le paramètre *M* correspond au nombre maximal de sommes de produits. Il est égale à 8.

Les ports du circuit sont décrits comme suit :

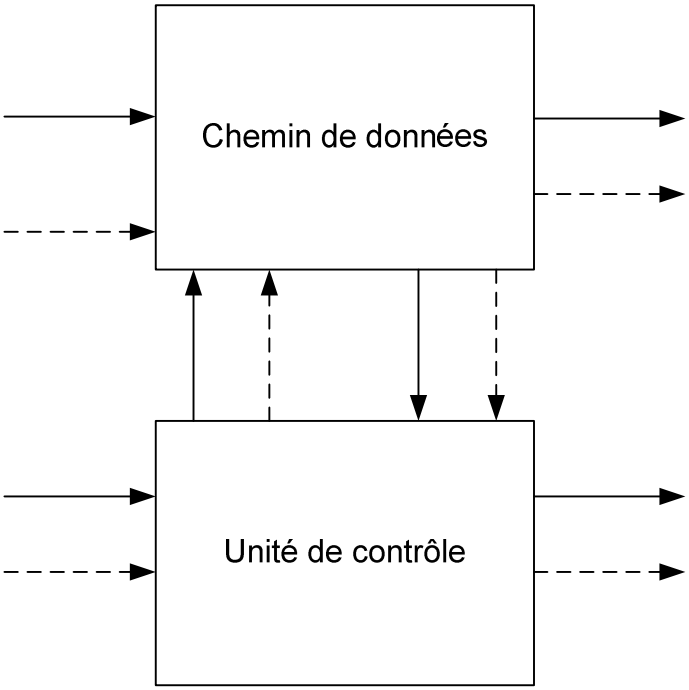
Nom	**Description**
Reset	Réinitialise le circuit.
CLK	L'horloge du système.
Go	Il est activé durant un coup d'horloge pour signifier le début des calculs.
Entreevalide	Il est activé en même temps que go, mais il reste actif à '1' durant l'envoi des entrées. En sachant que *M* données peuvent être envoyées au maximum, ainsi le nombre de donéées envoyées sera compris entre 1 et M.
Sortievalide	Ce port est activé durant un coup d'horloge pour signifier la fin des calculs et que la valeur sur le port *sortie* peut être lue.
Erreur	Le circuit ne peut calculer qu'une somme de *M* produits. Si le nombre d'entrées envoyées est supérieur à *M*, alors ce port sera activé pour indiquer qu'une erreur est survenue durant le calcul.
Sortie	La somme des produits.

Le processeur à but spécifique effectuera la somme des produits en sachant que le processeur reproduira l'algorithme suivant :

```
TantQue (TRUE)
        Attendre go;
        sortie = 0;
        i = 0;
    TantQue (entreevalide='1' ET i<M)
      Sortie = sortie + (entree x coef[i]);
      i++;
    Fin TantQue
    Si entreevalide='1' ET i== M Alors
      erreur = 1
      sortievalide = 0
    Sinon
      erreur = 0
      sortievalide = 1
    Fin Si
Fin TantQUE
```

Pour concevoir le processeur à but spécifique, vous devez :

a) Compléter le diagramme de l'architecture simplifiée de votre circuit en montrant les signaux et ports qui sont utilisés par le chemin de données et l'unité de contrôle. (2 points)

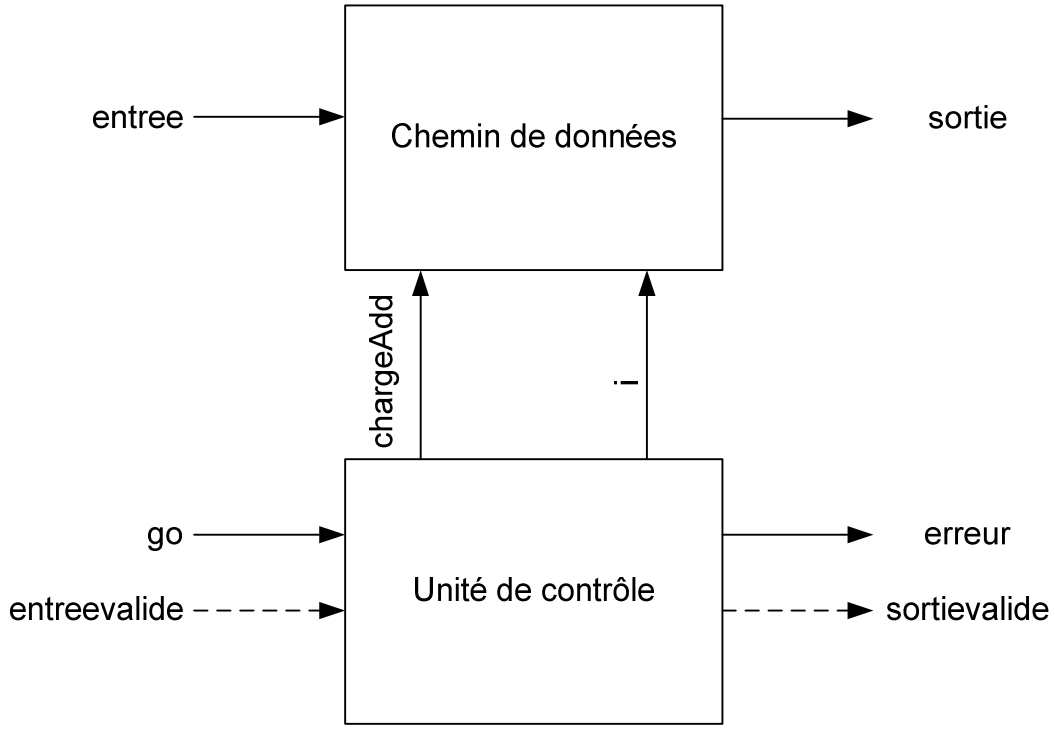


b) Donner le diagramme de la machine à états de l'unité de contrôle ainsi que le code VHDL de cette dernière. (6 points)

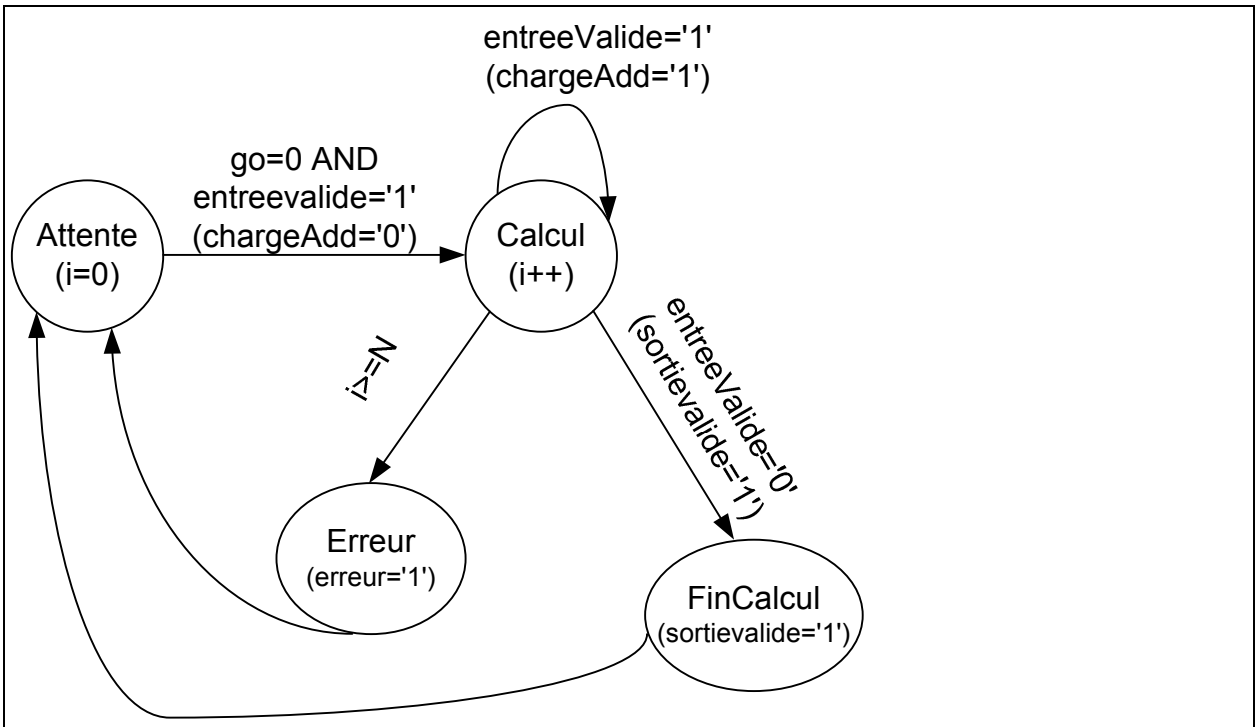
c) Donner le diagramme du chemin de données de votre circuit en sachant que la table des coefficients contient 8 cases. (4 points)

Solution

a. le diagramme de l'architecture simplifiée



b. la machine à états



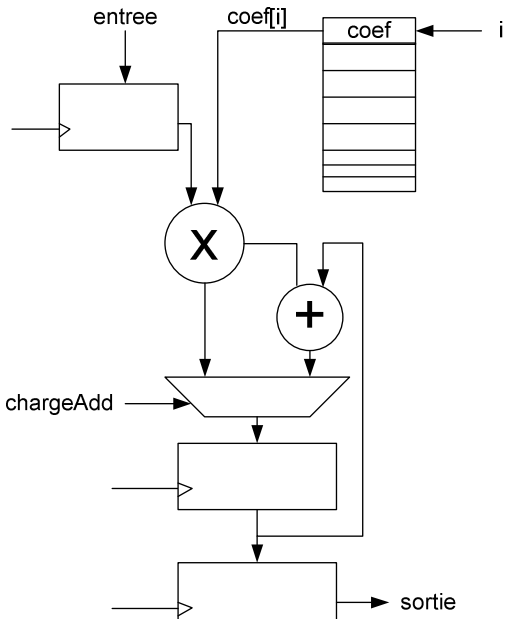
Code VHDL complet du processeur à but spécifique:

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

entity sommeproduit is
  generic (
    W : positive);
  port (
    reset        : in  std_logic;
    CLK          : in  std_logic;
    go : in  std_logic;
    entreevalide : in  std_logic;
    entree : in unsigned(W-1 downto 0);
    sortievalide : out std_logic;
    erreur       : out std_logic;
    sortie       : out unsigned(w-1 downto 0));

end sommeproduit;

architecture beg of sommeproduit is

  type state_t is (INIT, CALCUL, ER, FIN);
  signal cur_state : state_t;
  signal entree_t : unsigned(w-1 downto 0);
  signal charge_somme : std_logic;
  signal sommeprod : unsigned(W-1 downto 0);
  type coef_t  is array (integer range <>) of unsigned(W-1 downto 0) ;
```

```
 constant coef : coef_t (0 to 7)  := (X"01", X"02", x"03", x"04", x"05", x"06", x"07", x"08");
 signal i  : integer range 8 downto 0;
begin  -- beg

 unite_controle: process (CLK, reset)
 begin  -- process unite_controle
  if reset = '1' then               -- asynchronous reset (active low)
    cur_state <= INIT;
    charge_somme <= '0';
    i <= 0;
    sortievalide <= '0';
    erreur <= '0';

  elsif CLK'event and CLK = '1' then  -- rising clock edge
    charge_somme <= '0';
    sortievalide <= '0';
    erreur <= '0';

    case cur_state is
      when INIT =>
        cur_state <= INIT;
        i <= 0;
        if (go = '1') then
          cur_state <= CALCUL;
        end if;
      when CALCUL =>
        if (i>=7)  AND entreevalide ='1' then
          cur_state <= ER;
        elsif entreevalide='0' then
          cur_state <= FIN;
        else
          cur_state <= CALCUL;
          charge_somme <= '1';
          i <= i + 1;
        end if;
      when ER =>
              cur_state <= INIT;
              erreur <= '1';
      when FIN =>
               cur_state <= INIT;
               sortievalide <= '1';
      when others =>
        cur_state <= INIT;
        i <= 0;
    end case;
  end if;
end process unite_controle;

 chemin_donnees: process (CLK, reset)
 begin  -- process chemin_donnees
  if reset = '1' then               -- asynchronous reset (active low)
```

```
    sommeprod <= (others=>'0');
    entree_t <=(others=>'0');
    sortie <= (others=>'0');
  elsif CLK'event and CLK = '1' then  -- rising clock edge
    entree_t <= entree;
    if charge_somme = '0' then
      sommeprod <= resize(entree_t * coef(i), 8);
    else
      sommeprod <= resize(sommeprod + (entree_t * coef(i)), 8);
    end if;
    sortie <= sommeprod;
  end if;
 end process chemin_donnees;
end beg;
```

c. Le diagramme du chemin de données



Question 2. (8 points)- Processeur à but général

Soit le processeur à but général présenté dans les notes de cours au chapitre 8. Nous désirons rajouter une instruction dans le jeu d'instructions de ce processeur. Cette nouvelle instruction effectue l'opération MAC-multiply and accumulate qui multiplie deux opérandes (Opin1 x Opin2) et le résultat de la multiplication est additionné à une troisième opérande (Opinout). Le résultat est ensuite écrit dans l'opérande Opinout.

Le code de l'instruction est sur 4 bits. Les opérandes Opinout et Opin1 donnent l'adresse de deux registres. L'adresse des registres est codée sur 4 bits. Opin2 correspond à une valeur constante qui est contenue sur 16 bits. Donc, l'instruction doit effectuer :

Opinout = Opinout + (Opin1 x Opin2)

a. Modifiez le diagramme du chemin de données du processeur. (3 points)

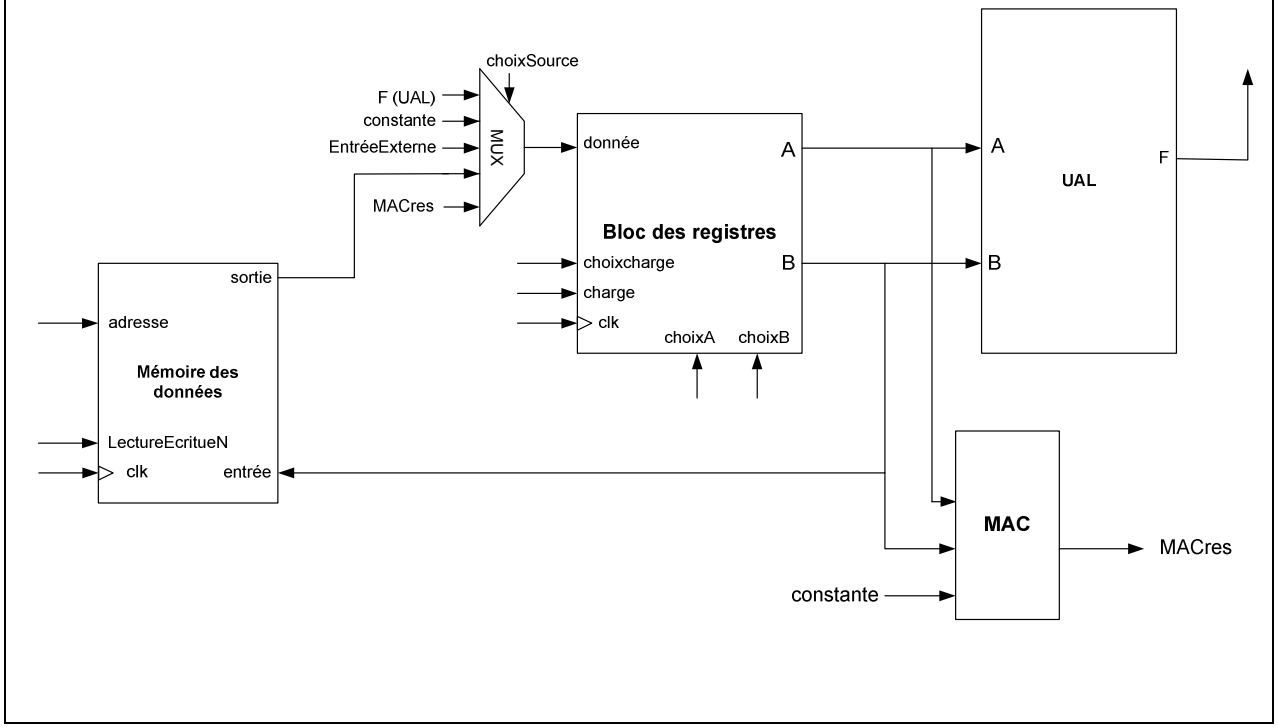
b. Modifiez le diagramme de la machine à états de l'unité de contrôle du processeur. N'oublier pas de préciser les valeurs des signaux de contrôle qui sont affectés par vos modifications. (3.5 points)

c. Donnez la liste et une brève description du/des signal/signaux signaux que vous rajoutez au processeur. Précisez aussi les modifications effectuées sur certains signaux existants de l'architecture. (1.5 point).

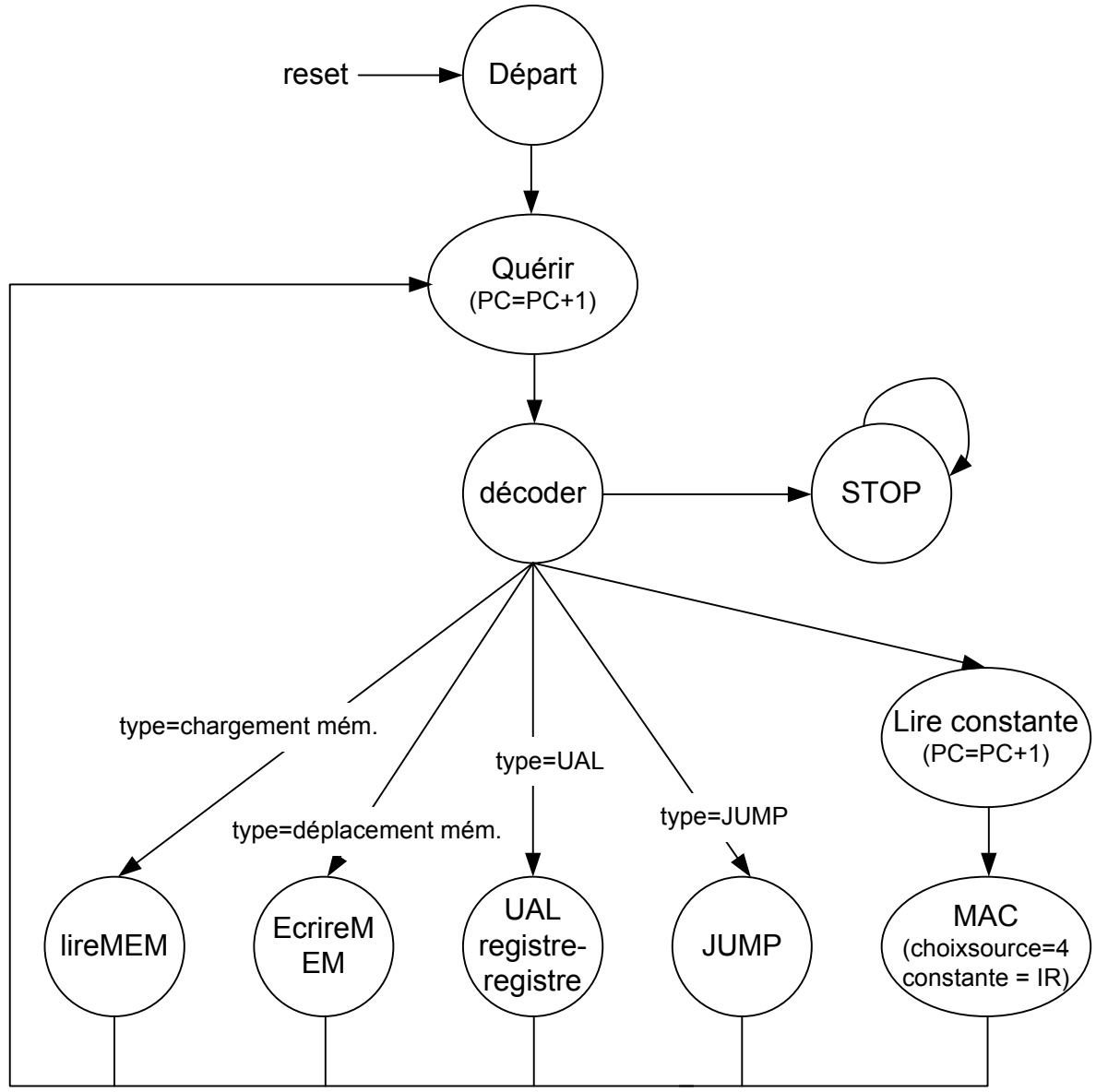
Détachez et remettez la page réponse dans votre cahier d'examen.

Solution

a. Diagramme du chemin de données



b. Diagramme d'états



c. La liste des signaux modifiés ou rajoutés

- choixSource : sa taille sera maintenant égale à 3 bits
- MACres : il conduit le résultat de la multiplcation-addition
- constante : sa taille est maintenant égale à 16 bits.

d.

INF3500-final H2009 – Question 2

Nom : **Matricule :**

Diagramme du chemin des données d'un processeur à usage général

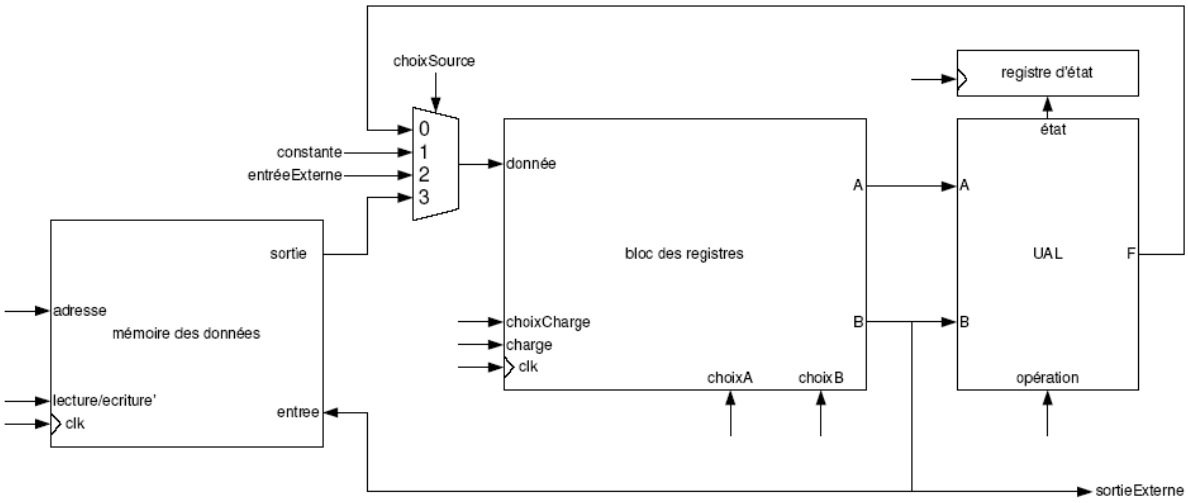
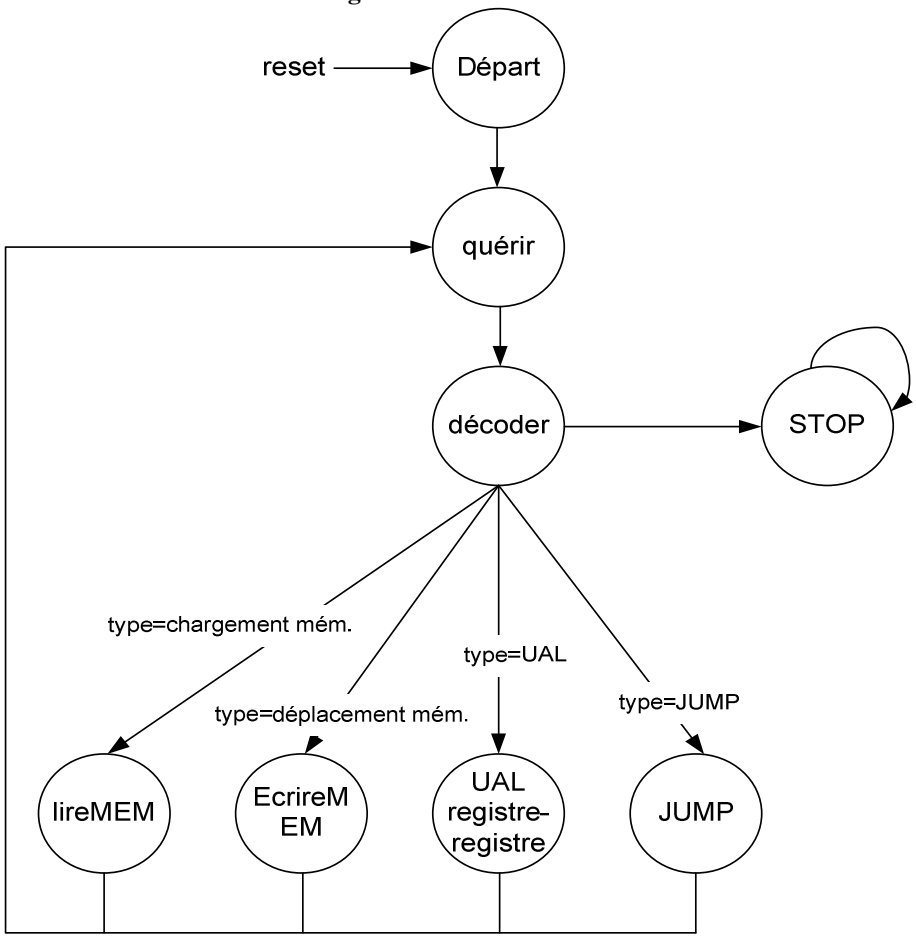


Diagramme d'états de l'unité de contrôle à modifier



Question 3. (6 points) – Circuit séquentiel

Concevez une pile de type LIFO (last in first out). Cette pile stocke des données et celles-ci seront dans l'ordre inverse de leur écriture à savoir que la dernière donnée écrite sera la première donnée lue.

La déclaration de l'entité de la LIFO est définie comme suit :

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

entity lifo is
    generic (
        N : positive := 4
        W : positive := 8
    );
    port (
        clk : in std_logic;
        reset : in std_logic;
        din : in std_logic_vector(W - 1 downto 0);
        dout : out std_logic_vector(W - 1 downto 0);
        wr_en : in std_logic;
        rd_en : in std_logic
        empty : out std_logic;
        full : out std_logic
    );
end fifo;
```

Les paramètres N et W représentent respectivement le nombre maximum d'éléments dans la pile et la largeur des données en bits.

Les ports de l'entité se décrivent comme suit :

- Le signal `clk` est un signal d'horloge synchronisant toutes les activités de la pile. Le signal `reset` permet de vider la pile.
- Les ports `din` et `dout` sont respectivement l'entrée et la sortie de la pile.
- Les ports `empty` et `full` indiquent respectivement que la pile est vide ou bien qu'elle est pleine.
- Quand le signal `wr_en` (*write enable*) est activé, la valeur placée sur le port `din` sera insérée dans la pile lors de la prochaine transition active du signal d'horloge `clk`.
- Quand le signal `rd_en` (*read enable*) est activé, la valeur insérée le plus récemment sur la pile sera écrite sur le port `dout` lors de la prochaine transition active du signal d'horloge `clk`.
- Il doit être possible d'écrire une nouvelle valeur dans la pile et de lire la valeur la plus récente simultanément.
- Lorsque la LIFO est pleine les requêtes d'écriture doivent être ignorées et lorsque la LIFO est vide les opérations de lecture doivent être ignorées.

Donnez le code VHDL de l'architecture de la LIFO.

Solution

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

entity lifo is
  generic (
    N : positive := 4; -- la profondeur (le nombre d'éléments) dans la pile
    W : positive := 8 -- la largeur (en bits) de la pile
    );
  port (
    clk : in std_logic;
    reset : in std_logic;    -- actif haut: un '1' réinitialise la pile
    din : in std_logic_vector(W - 1 downto 0); -- données entrant dans la pile
    dout : out std_logic_vector(W - 1 downto 0); -- données sortant de la pile
    wr_en : in std_logic; -- write-enable: si actif, une donnée sera lue de din et
    -- entrée dans la pile au prochain front montant de clk
    rd_en : in std_logic; -- read-enable: si actif, une donnée sera sortie de la pile
    -- et placée sur dout au prochain front montant de clk
    empty : out std_logic; -- indique que la file est vide
    full : out std_logic -- indique que la file est pleine
    );
end lifo;

architecture tache of lifo is
  type List is array (NATURAL range <>) of std_logic_vector(W - 1 downto 0);
  signal FIFOList : List(N - 1 downto 0);
  signal writeptr : integer range 0 to N-1 := 0;

  signal plein :  std_logic ;
  signal vide :  std_logic;

begin
  process(clk, reset) -- Intervient au coup d'horloge
    variable nb_elem : integer range N-1 downto 0;
  begin
    if reset ='1' then
      writeptr <= 0;
      vide <= '1';
      plein <= '0';
      dout <= (others => '0');
    elsif rising_edge(clk) then
      if rd_en = '1' and vide = '0' then        --Mode Lecture (rd et pas vide)
        dout <= FIFOList(writeptr-1);        --lit à la position du pointeur
        plein <= '0';
        nb_elem := nb_elem -1;
        if nb_elem = 0 then
          vide <= '1';                    --liste vide si les 2 pointeurs sont égaux après lecture
        end if;
      end if;

      if wr_en = '1' and plein = '0' then     --Mode écriture (wr et pas plein)
```

```
      FIFOList(writeptr) <= din;  --écrit à la position du pointeur
      nb_elem := nb_elem + 1;

      if writeptr >= N - 1 then
        writeptr <= 0;
      else
        writeptr <= writeptr + 1;    --déplace le pointeur
      end if;
      vide <= '0';
      if(nb_elem = N) then
        plein <= '1';
      end if;
    end if;
  end if;
end process;

 empty <= vide;
 full <= plein;
end tache;
```

Question 4. (4 points) – circuit combinatoire

Soit, le circuit d'un décaleur de bits vers la gauche *shift_Nbits*. Le circuit comprend :

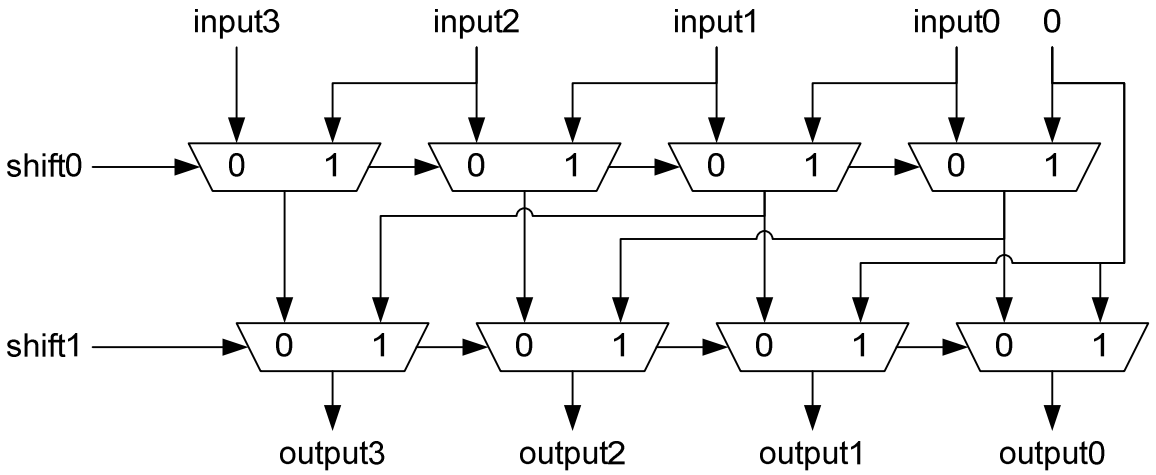
- un port d'entrée de 4 bits *input* de type *unsigned.*
- un port de contrôle de 2 bits *shift* qui indique combien de décalages doivent être effectués. Ce port est de type *std_logic_vector*.
- un port de sortie de 4 bits *output* de type *unsigned*.

Par exemple :

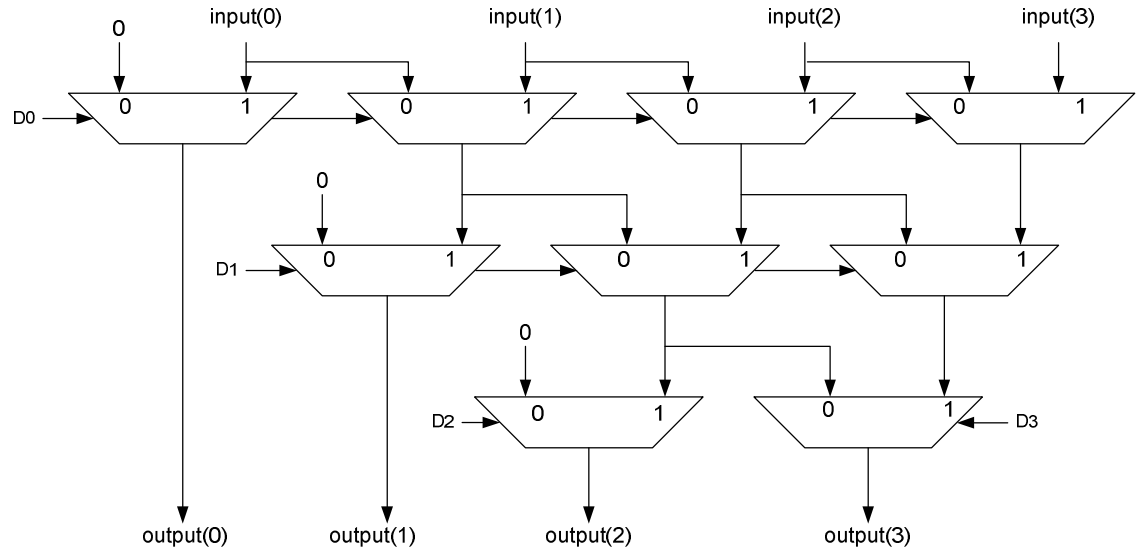
Si le port *input* est égal à "1010" et *shift* est égal à "10" alors *output* sera égal à "1000".

Donnez le diagramme du circuit d'un décaleur vers la gauche de 4 bits. Vous pouvez utiliser des opérations logiques et arithmétiques, la comparaison, des multiplexeurs, décodeurs et encodeurs, éléments mémoire, etc. Vous ne pouvez pas utiliser l'opération shift.

Première Solution



Deuxième Solution (fonctionne, mais nécessite plus de ressources)



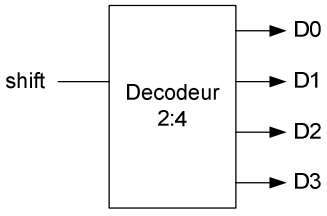


Table de vérité du décodeur :

Shift	D3	D2	D1	D0
00	1	1	1	1
01	1	1	1	0
10	1	1	0	0
11	0	0	0	0

*** Pour illustration seulement (pas demandé dans la question) ***

Code VHDL, deux solutions proposées ainsi que le banc d'essai

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

entity shifter is
  port (
    shift : in  std_logic_vector(1 downto 0);
    input : in unsigned(3 downto 0);
    output       : out unsigned(3 downto 0));

end shifter;

architecture beh1 of shifter is
begin  -- beh
  process (shift, input)
      variable sortie_t : unsigned(3 downto 0);
  begin  -- process
    sortie_t := input;
    for i in 0 to 3 loop
      if (i=to_integer(unsigned(shift))) then
        exit;
      end if;
      sortie_t := sortie_t SLL 1;
    end loop;  -- i

    output <= sortie_t;
  end process;
end beh1;


architecture beh2 of shifter is
begin  -- beh
 output <=  input sll to_integer(unsigned(shift));
end beh2;


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

entity shifter_tb is
end shifter_tb;

architecture beh1 of shifter_tb is
component shifter
  port (
```

```
    shift : in  std_logic_vector(1 downto 0);
    input : in unsigned(3 downto 0);
    output       : out unsigned(3 downto 0));
end component;
    signal shift :   std_logic_vector(1 downto 0);
    signal input :  unsigned(3 downto 0);
    signal output :  unsigned(3 downto 0);
begin  -- beh
  uut : shifter
    port map (
      shift, input, output);

  input <= X"1";
  process
  begin  -- process
    for i in 0 to 3 loop
      shift <= std_logic_vector(to_unsigned(i,2));
      wait for 10 ns;
    end loop;  -- i
  end process;
end beh1;
```

Question 5. (6 points) – questions brèves

a. Rappelez les différentes phases du flot de conception qui mènent au déploiement d'un circuit sur un FPGA. Indiquez quelle est la phase la plus importante du flot et expliquez pourquoi. (1.5 points)

b. Énumérez les méthodes de vérification et pour chaque méthode citez un type d'application approprié à l'utilisation de la méthode. (2 points)

c. Donnez les différentes causes de déphasage d'horloge d'un circuit. (1.5 points)

e. Énumérez quatre considérations d'implémentation dont il faut tenir compte durant la conception d'un circuit numérique. (1 point)

Solution

a. conception et génération du code HDL, vérification, synthèse, implémentation, génération du fichier de configuration.

La vérification est la phase la plus importante du processus de conception. Cette dernière garantit que la fonctionnalité du circuit est respectée et c'est l'objectif primaire à atteindre avant de passer aux étapes de synthèse et d'implémentation.

b. Méthodes de vérification

- par tableaux de constantes : quand le nombre de valeurs à tester est limité et doit être fixe.
- aléatoire : algorithmes mathématiques, conversion de type pouce à mètre par exemple.
- fichiers : pour des applications de traitement d'images.
- exhaustive : des machines à états

c.

- un mauvais routage du signal d'horloge avec des chemins de longueur différente;
- un mauvais routage du signal d'horloge avec un débalancement de la charge du signal; et,
- l'introduction de signaux de contrôle du signal d'horloge avec de la logique combinatoire (*clock gating*).

e. les considérations d'implémentation :

- Précision des calculs
- Puissance consommée
- Taille du système (en nombre de portes logiques ou transistors par exemple)
- Taux de traitement (nombre d'opérations traités par unité de temps)

Question 6. (4 points) – Implémentation avec la technologie PAL

Soit le code VHDL suivant :

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

entity question4 is
  port (
    clk : in std_logic;
    x1 : in std_logic;
    x2 : in std_logic;
    x3 : in std_logic;
    y1 : out std_logic;
    y2 : out std_logic;
    );
end question4;

architecture beh of question4 is
  signal p1 : std_logic;
  signal p2 : std_logic;
  signal p3 : std_logic;
  signal p4 : std_logic;
begin  -- beh

  p1 <= x1 and x2 and not x3;
  p2 <= not x1 and x2 and x3;
  p3 <= not x1 and not x2;
  p4 <= x1 and not x2 and not x3;

  process (clk, <reset name>)
  begin  -- process
    if clk'event and clk = '1' then  -- rising clock edge
      y1 <= p1 or p2;
    end if;
  end process;

  y2 <= p3 or p4;
end beh;
```

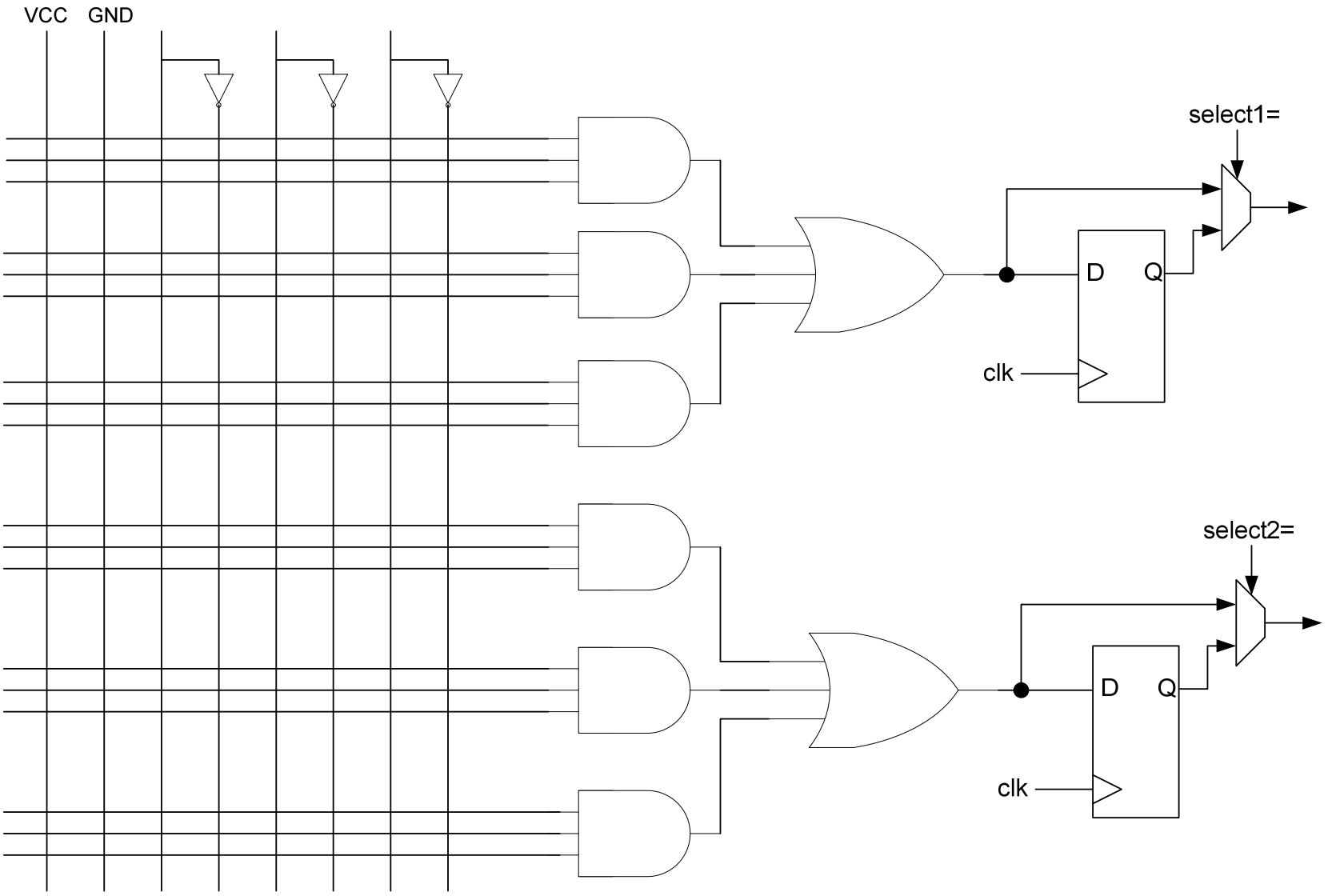
Complétez le circuit PAL suivant qui implémentera la fonctionnalité décrite dans le code VHDL du circuit *question4*. N'oubliez pas de définir la valeur des signaux de programmation des multiplexeurs (select1 et select2).

Détachez et remettez la page réponse dans votre cahier d'examen.

Ce circuit PAL comporte des bascules qui peuvent être utilisées selon la valeur du signal de contrôle du multiplexeur. Si le signal select est égal à '1' alors la valeur de Q sera conduite sur la sortie du multiplexeur. Sinon la sortie du OR sera multiplexée sur la sortie du multiplexeur.

INF3500-final H2009 – Question 6

Nom : **Matricule :**



Solution