INF3500 : Conception et réalisation de systèmes numériques

Examen intra

Mercredi 22 juin 2011

Durée: 2h.
Pondération: 20%.
Documentation: Une feuille recto verso 8.5"×11" ou A4 permise.
Calculatrice: Programmable permise.
Directives particulières:

- Ordinateurs interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre sur le questionnaire et le remettre.
- Ne posez pas de questions. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.

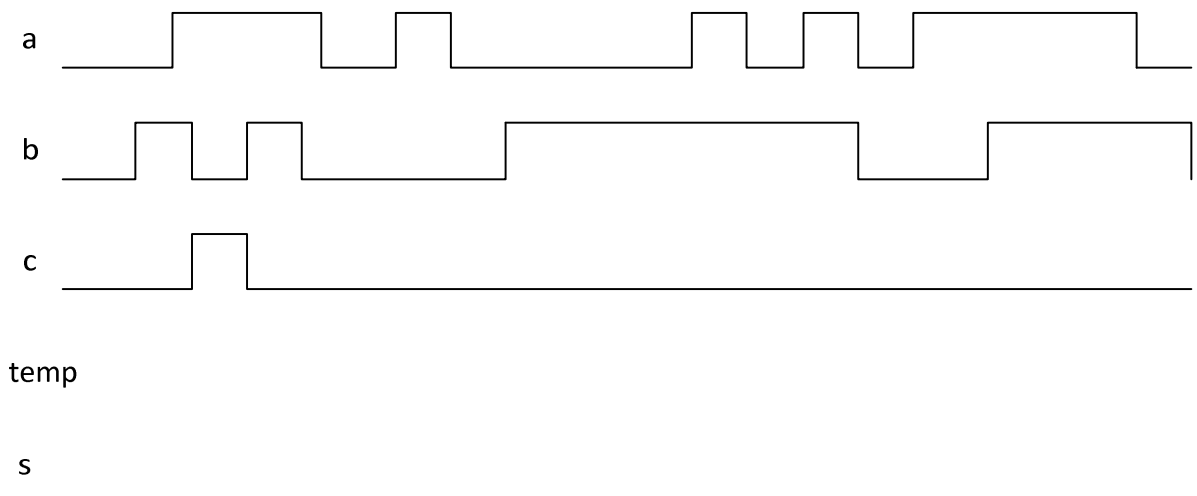
Question 1. (3 points)

Considérez le code vhdl suivant.

```
signal a: std_logic;
signal b : std_logic;
signal c: std_logic;
signal s: std_logic:='0';

process(a,b,c)
variable temp: std_logic;
begin
   temp:= b or c;
   temp:= temp and c;
   s <= temp and not a;
   s <= s or not a;
end process
```

a. Donnez la différence entre un signal et une variable.(1 point)

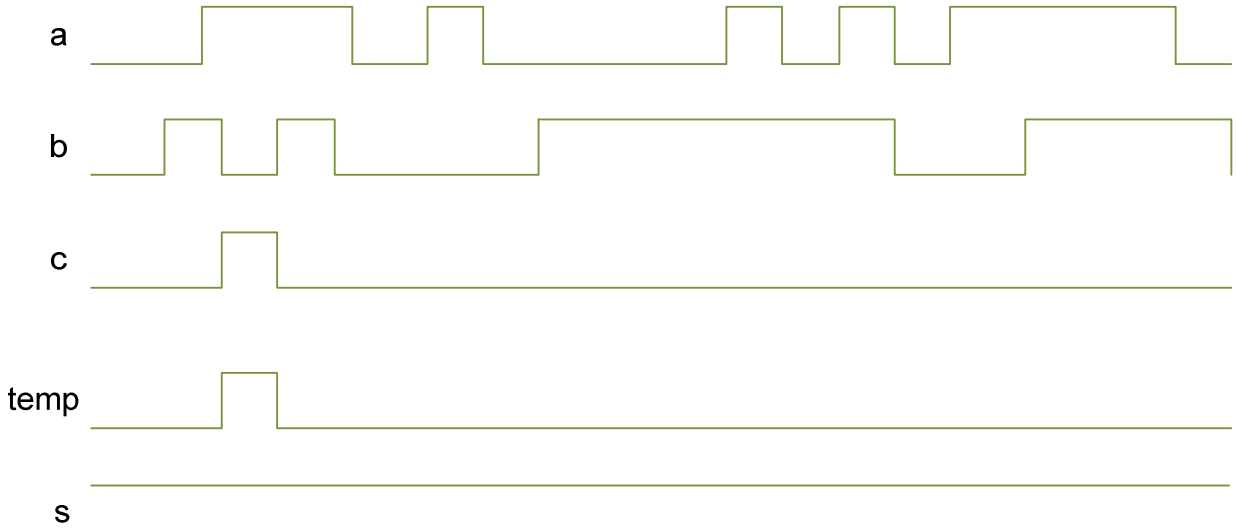
b. En vous servant du code donné, complétez le diagramme temporel suivant (2 points)



Solution :

a. Voir notes de cours, section 2.3.3

b.



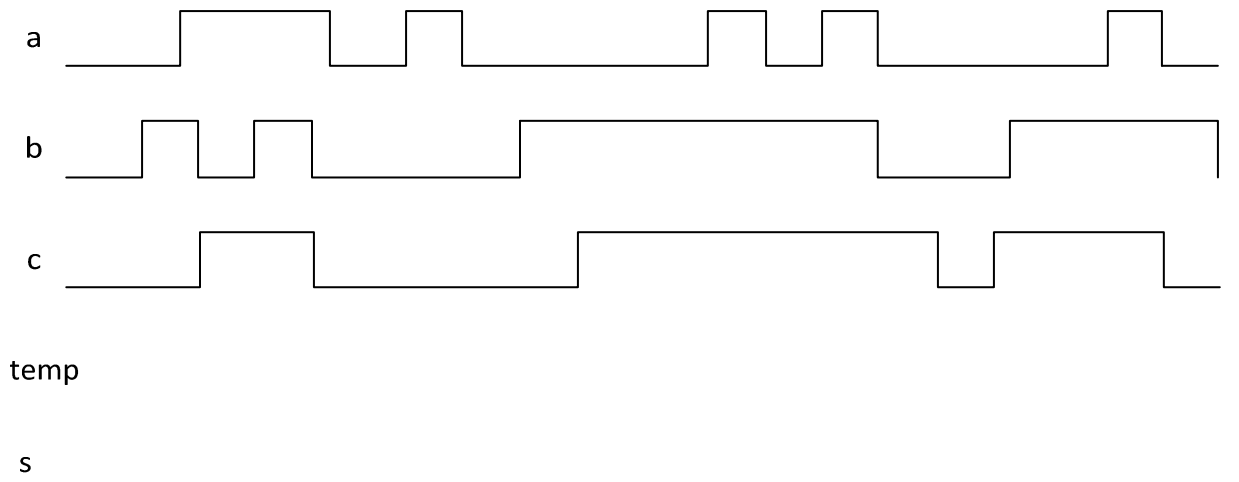
Question 2. (3 points)

Considérez le code vhdl suivant.

```
signal a: std_logic;
signal b : std_logic;
signal c: std_logic;
signal s: std_logic;

process(a,b,s)
variable temp: std_logic;
begin
   temp:= b and c;
   s <= temp and not a;
end process
```

a. Expliquez brièvement comment le simulateur évalue un process en fonction de la liste de sensibilité. (1 point)

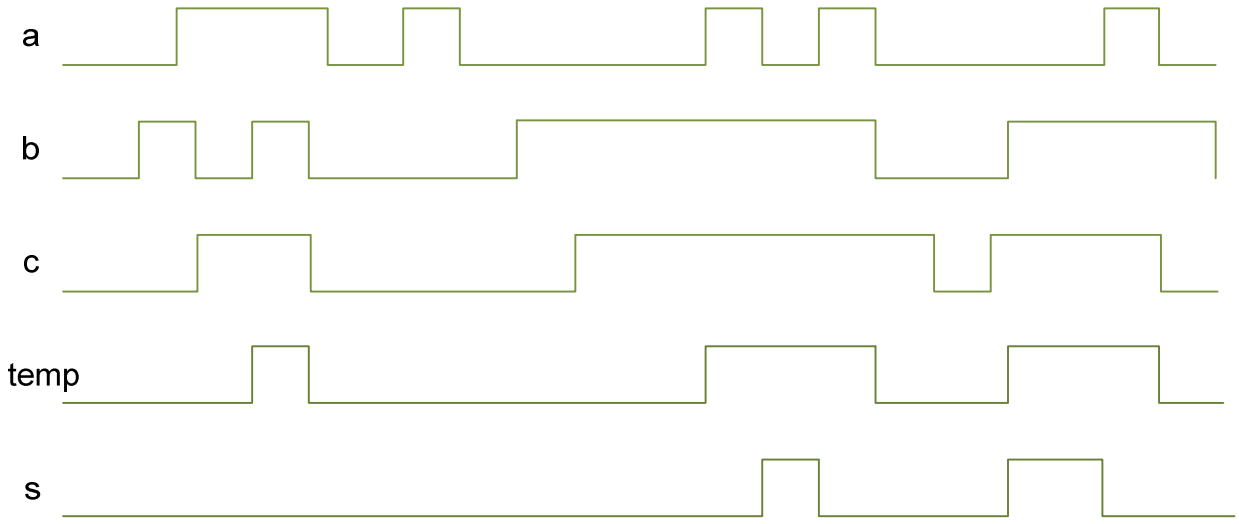
b. En vous servant du code donné, complétez le diagramme temporel suivant : 2 points



Solution :

a. Voir notes de cours, section 7.3.1

b.



Question 3. (4 points)

Considérez le code VHDL suivant. Le circuit accepte en entrée un nombre non signé encodé sur 10 bits, et il a trois sorties. Ces sorties donnent le nombre de centaines, de dizaines et d'unités (de 0 à 9 inclusivement,
encodées sur 4 bits) du nombre. La sortie erreur est activée si le nombre est égal ou supérieur à 1000.

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

entity unsigned2dec is
port(
   nombre : in unsigned(9 downto 0);
   centainesBCD, dizainesBCD, unitesBCD : out unsigned(3 downto 0);
```

```
   erreur : out std_logic
);
end unsigned2dec;

architecture arch of unsigned2dec is
begin

   erreur <= '1' when nombre >= 1000 else '0';
   process(nombre)
   variable n, c, d, u : natural := 0;
   begin
      n := to_integer(nombre);
      for centaines in 9 downto 1 loop
         c := 0;
         if n >= centaines * 100 then
            c := centaines;
            exit;
         end if;
      end loop;
      n := n - c * 100;
      for dizaines in 9 downto 1 loop
         d := 0;
         if n >= dizaines * 10 then
            d := dizaines;
            exit;
         end if;
      end loop;
      u := n - d * 10;
      centainesBCD <= to_unsigned(c, 4);
      dizainesBCD <= to_unsigned(d, 4);
      unitesBCD <= to_unsigned(u, 4);
   end process;

end arch;
```

Pour l'entité `unsigned2dec` décrite ci-dessus, composez un banc d'essai qui effectue un test exhaustif et qui vérifie que la sortie est valide en tout temps. Une sortie valide indique correctement les unités, les dizaines et les centaines pour des nombres entre 0 et 999, inclusivement. Pour les autres nombres, le signal d'erreur doit être actif. Dans le cas d'une erreur, votre banc d'essai doit simplement afficher le message « erreur » à la console et se terminer.

Complétez le code VHDL suivant en donnant l'architecture.

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

entity unsigned2dec_TB is
end unsigned2dec_TB;

architecture TB_ARCHITECTURE of unsigned2dec_TB is
--- votre code ici
begin
--- votre code ici
end TB_ARCHITECTURE;
```

Une solution possible:

```
architecture TB_ARCHITECTURE of unsigned2dec_TB is

signal nombre : unsigned(9 downto 0);
signal centainesBCD, dizainesBCD, unitesBCD : unsigned(3 downto 0);
signal erreur : std_logic;

begin

UUT : entity unsigned2dec(arch)
port map (nombre, centainesBCD, dizainesBCD, unitesBCD, erreur);

process
variable n : natural;
```

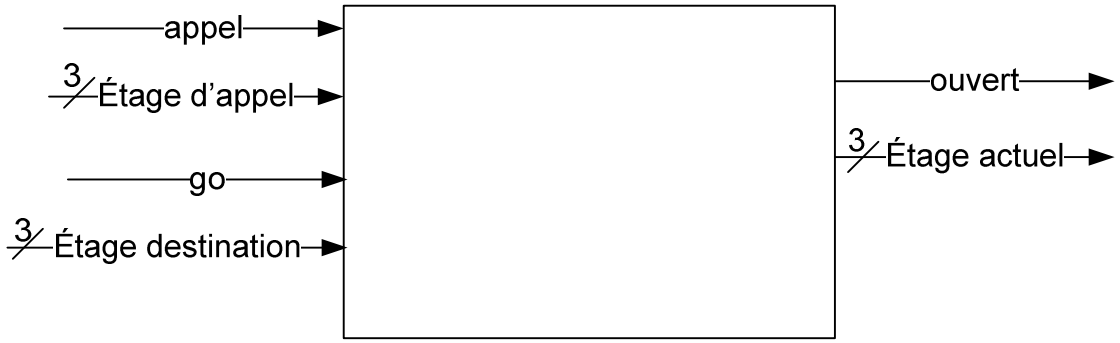
```
begin

for k in 0 to 1023 loop
   nombre <= to_unsigned(k, nombre'length);
   wait for 10 ns;
   n := to_integer(centainesBCD) * 100
   + to_integer(dizainesBCD) * 10
   + to_integer(unitesBCD);
   if (k <= 999) then
      assert erreur = '0' report "erreur!" severity failure;
      assert to_integer(centainesBCD) <= 9 report "erreur!" severity failure;
      assert to_integer(dizainesBCD) <= 9 report "erreur!" severity failure;
      assert to_integer(unitesBCD) <= 9 report "erreur!" severity failure;
      assert n = k report "erreur!" severity failure;
   else
      assert erreur = '1' report "erreur!" severity failure;
   end if;
end loop;
report "simulation terminée" severity failure;
end process;

end TB_ARCHITECTURE;
```

Question 4. (10 points)

On vous demande de concevoir le chemin de données et la machine à état d'un ascenseur servant dans un immeuble à 8 étages (allant de l'étage 0.à l'étage 7).

L'ascenseur peut être dans quatre états possibles :

- Il peut être en état d'attente d'une nouvelle commande. En état d'attente, il se rend à l'étage d'appel lorsque l'utilisateur l'appelle (signal « appel » mis à 1)
- Il peut être ouvert (pour accepter un nouveau passager ou en faire sortir un). Quand il est ouvert, l'utilisateur peut rentrer l'étage désiré et le valider avec le signal « go »
- Si après 3 secondes d'ouverture, l'utilisateur ne rentre pas l'étage désiré, l'ascenseur retourne en mode d'attente
- Il peut être en train de monter pour se rendre à l'étage désiré ou l'étage d'appel
- Il peut être en train de descendre pour se rendre à l'étage désiré ou l'étage d'appel



Les entrées/sorties de l'entité sont les suivantes :

Appel : actif quand un passager veut utiliser l'ascenseur

Étage d'appel : étage où se situe le passager qui veut utiliser l'ascenseur

Étage destination : une fois entré dans l'ascenseur, le passager sélectionne l'étage où il veut se rendre

Go : actif quand l'utilisateur a rentré l'étage de destination

Ouvert : actif quand l'ascenseur est ouvert

On suppose que ca prend une seconde pour se rendre d'un étage à l'autre. Une horloge de 1 Hz est utilisée pour ce circuit

a. En fonction du degré d'abstraction désiré et du type d'opération à effectuer, discutez du type de signal que vous pouvez utiliser pour concevoir ce système. (1 point)

b. Dessiner la machine à état du système. (4 points)

c. Décrire la machine à état à l'aide du code VHDL. Les signaux de contrôle peuvent être implicites. (5 points)

Solution :

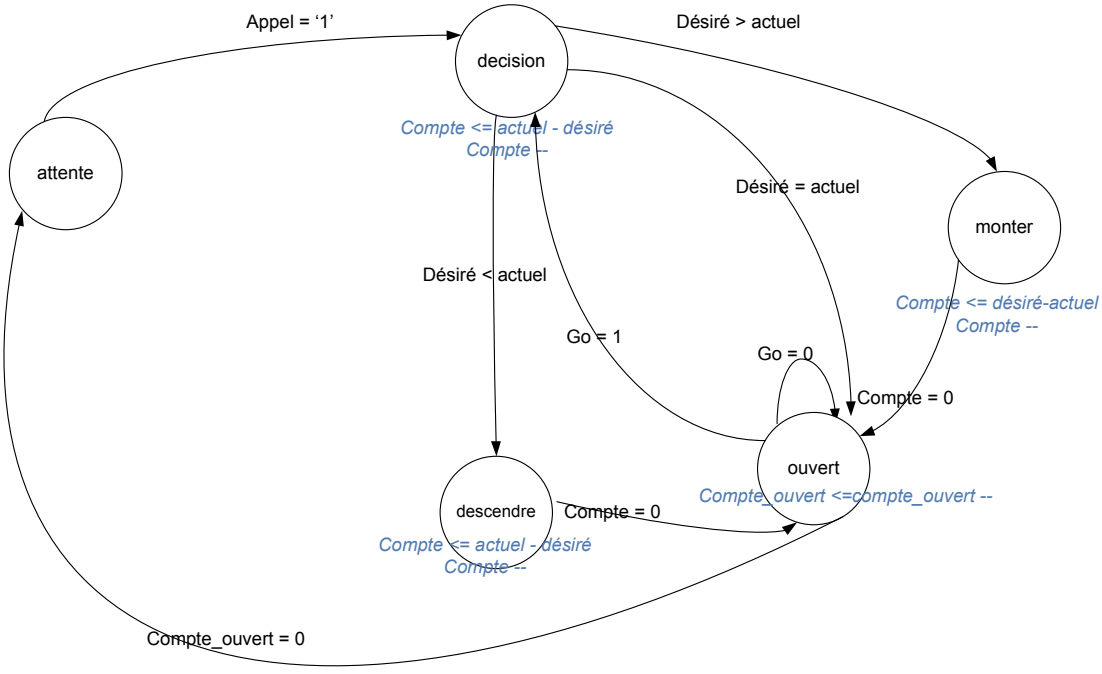
a. Le type std_logic_vector (librairie .std_logic_1164) pourrait être utilisé s'il n'y a pas d'opération arithmétique à effectuer. Dans notre cas, le compteur effectue une opération arithmétique.

Le type unsigned (librairie numeric_std) permet d'effectuer des opération arithmétiques, et dérive directement du type std_logic

Le type integer est plus abstrait mais il ne faut pas oublier de spécifier la largeur des bits.

Le type unsigned est le bon compromis

b.



c.

```
-- Code VHDL
library IEEE;
```

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

entity ascenseur is
port(
   reset_n            : in std_logic;
   clk                : in std_logic;
   appel              : in std_logic;
   etage_appel        : in std_logic_vector(2 dowmto 0);
   go                 : in std_logic;
   etage_destination  : in std_logic_vector(2 downto 0);
   ouvert             : out std_logic;
   etage_actuel       : out std_logic_vector(2 downto 0)
);
end ascenseur;

architecture arch of ascenseur is
   signal compte: unsigned(2 downto 0);
   signal compte_ouvert: unsigned(1 downto 0);
   signal reg_actuel : unsigned(2 downto 0);
   signal reg_desire: unsigned(2 downto 0);
   type t_etat is (attente, decision, ouverture, monter, descendre);
   signal etat : t_etat;

begin


process(clk,reset_n)
begin
  if reset_n = '0' then
     etat          <= attente;
     ouvert        <= '0';
     compte        <= (others => '0');
    compte_ouvert <= (others => '0');
    reg_actuel    <= (others => '0');
    reg_desire    <= (others => '0');
  elsif clk'event and clk = '1' then
      case etat is
         when attente =>
            if appel = '1' then
               reg_desire <= unsigned(etage_appel);
               etat       <= decision;
            end if;
         when decision =>
            if reg_desire = reg_actuel then
             etat   <= ouverture;
             ouvert <= '1';
             compte_ouvert <= "10";
          elsif reg_desire > reg_actuel then
             etat   <= descendre;
             ouvert <= '0';
             compte <= reg_actuel - reg_desire;
          elsif reg_desire < reg_actuel then
             etat <= monter;
             ouvert <= '0';
             compte <= reg_desire - reg_actuel;
          end if;
        when monter =>
            compte <= compte - 1;
          if compte = "000" then
             etat   <= ouverture;
             ouvert <= '1';
             compte_ouvert <= "10";
          else
               reg_actuel <= reg_actuel + 1;
             compte <= compte - 1;
               ouvert <= '0';
```

```
          end if;
        when descendre =>
            compte <= compte - 1;
          if compte = "000" then
             etat   <= ouverture;
             ouvert <= '1';
             compte_ouvert <= "10";
          else
               reg_actuel <= reg_actuel - 1;
             compte <= compte - 1;
               ouvert <= '0';
          end if;
         when ouverture =>
           ouvert <= '1';
           if go = '0' then
              compte_ouvert <= compte_ouvert - 1;
              if compte_ouvert = "00" then
                  etat <= attente;
               ouvert <= '0';
              end if;
           else
              reg_desire <= unsigned(etage_destination);
              etat <= decision;
            ouvert <= '0';
           end if;
        when others => etat <= attente;
    end case;
end if;
end process;

etage_actuel <= std_logic_vector(reg_actuel);

end arch;
```