INF3500 : Conception et réalisation de systèmes numériques

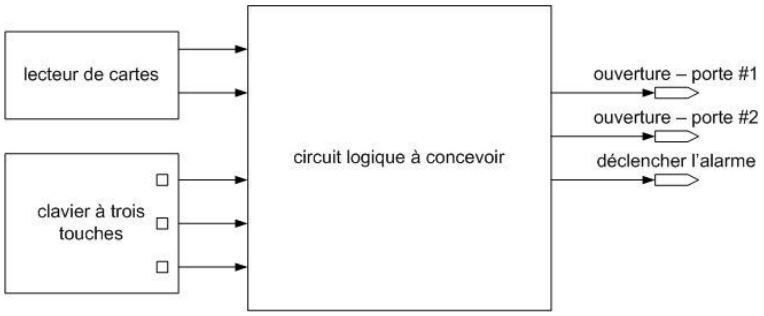
Examen intra

18 juin 2007

Durée: 2h.
Pondération: 20%.
Documentation: Toute 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. (4 points)

Un système simple de sécurité pour un immeuble consiste en un lecteur de cartes magnétiques, d'un clavier à trois touches, de deux portes et d'une alarme.



Un utilisateur doit entrer son code d'identification personnel en pesant sur une, deux ou trois touches du clavier simultanément, puis glisser sa carte dans le lecteur. L'alarme est déclenchée si la carte n'est pas reconnue, ou si un code incorrect est entré pour la carte glissée. La porte correspondante est débarrée si la carte est valide et que le bon code est entré.

Le lecteur de cartes donne les codes suivants selon la carte glissée:

état	signal A	signal B
pas de carte glissée	0	0
carte conforme pour la porte #1	0	1
carte conforme pour la porte #2	1	1
carte non valide	1	0

Les codes d'identification personnels acceptables sont les suivants:

porte	codes valides
#1	101 et 110
#2	101 et 011

Donnez un modèle VHDL synthétisable d'un circuit combinatoire correspondant à ce système de sécurité.

Solution :

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity securite is
   port(
      A, B : in std_logic;
      code : in std_logic_vector(2 downto 0);
      porte1, porte2, alarme : out STD_LOGIC
   );
end securite;

architecture arch of securite is

begin

   process(A, B, code)
   variable carte1 : boolean := false;
   variable carte2 : boolean := false;
   variable carteNV : boolean := true;
   variable code1 : boolean := false;
   variable code2 : boolean := false;
   begin

      carte1 := (A = '0' and B = '1');
      carte2 := (A = '1' and B = '1');
      carteNV := (A = '1' and B = '0');

      code1 := (code = "101" or code = "110");
      code2 := (code = "101" or code = "011");

      if (carte1 and code1) then
         porte1 <= '1';
      else
         porte1 <= '0';
      end if;

      if (carte2 and code2) then
         porte2 <= '1';
      else
         porte2 <= '0';
      end if;

      if ( carteNV or (carte1 and not(code1)) or (carte2 and not(code2)) ) then
         alarme <= '1';
      else
         alarme <= '0';
      end if;

   end process;

end arch;
```

Question 2. (4 points)

Un circuit combinatoire a quatre entrées (R, S, T, U) et deux sorties (Y, Z). L'entrée RSTU représente un chiffre de 0 à 9 encodé avec le code BCD. Les deux bits YZ représentent le reste quand RSTU est divisé par 3. Vous pouvez supposer que seulement des entrées valides seront appliquées au système.

Donnez les équations booléennes pour les sorties Y et Z sous la forme d'une somme de produits minimale. (Ne pas donner de code VHDL)

Solution :

Y = S'TU' + ST'U + RU'

Z = ST'U' + R'S'T'U + STU

Question 3. (4 points)

La déclaration d'entité suivante en VHDL correspond à un circuit combinatoire qui accepte en entrée un nombre de six bits et qui a une sortie pour indiquer si le nombre est divisible par sept. Donnez une architecture pour cette entité.

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

entity multipleDeSept is
    port (
        I : in unsigned(5 downto 0);
        F : out std_logic
    );
end multipleDeSept;
```

Solution :

```
architecture arch of multipleDeSept is
begin
    with to_integer(I) select
        F <=
        '1' when 0 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63,
        '0' when others;
end arch;
```

Question 4. (4 points)

Complétez le banc d'essai VHDL suivant pour vérifier, de façon exhaustive, le circuit de la question 3. Le banc d'essai doit indiquer, à la console seulement, pour quels cas le circuit à vérifier donne une sortie incorrecte.

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

entity multipleDeSeptTB is
end multipleDeSeptTB;

architecture arch of multipleDeSeptTB is
```

```
component multipleDeSept
port (I : in unsigned(5 downto 0); F : out std_logic);
end component;

signal I : unsigned(5 downto 0);
signal F : std_logic;

begin

-- votre code ici

end arch;
```

Solution :

```
UUT : multipleDeSept port map (I, F);

process
constant kmax : integer := 63;
begin
    for k in 0 to kmax loop
        I <= to_unsigned(k, I'length);
        wait for 10 ns;
        assert ((k mod 7 = 0) = (F = '1'))
            report "erreur pour l'entrée " & integer'image(k) severity error;
        assert k < kmax
            report "simulation terminée" severity failure;
    end loop;
end process;
```

Question 5. (4 points)

Considérez le modèle VHDL suivant. Donnez le diagramme d'états correspondant.

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

entity controleQ5 is
    port (
        reset, CLK : in STD_LOGIC;
        x : in STD_LOGIC_VECTOR(1 downto 0);
        sortie : out STD_LOGIC
    );
end controleQ5;

architecture arch of controleQ5 is
type type_etat is (S1, S2, S3, S4);
signal etat : type_etat := S1;
begin

    process(CLK, reset) is
    begin
        if (reset = '0') then
            etat <= S1;
        elsif (rising_edge(CLK)) then
            case etat is
                when S1 =>
                    if x = "00" then
                        etat <= S3;
```

```
                elsif x = "01" then
                    etat <= S2;
                end if;
            when S2 | S3 =>
                if x = "10" then
                    etat <= S4;
                elsif x = "11" then
                    etat <= S1;
                end if;
            when S4 =>
                etat <= S1;
        end case;
    end if;
end process;

process(x, etat)
begin
    case etat is
        when S1 | S4 =>
            sortie <= '1';
        when S2 | S3 =>
            if x = "10" then
                sortie <= '0';
            else
                sortie <= '1';
            end if;
    end case;
end process;

end arch;
```

Solution :