INF3500 : Conception et réalisation de systèmes numériques

Examen intra

18 juin 2008

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 encodeur simplifié de code Morse prend en entrée un vecteur de 7 bits (InCar), encode ce dernier et fournit le code correspondant sur trois ports de sortie (OCar1, OCar2, OCar3) qui sont chacun des vecteurs de 2 bits. Ce module encode seulement les caractères ascii : A, D, G, I, K et M, comme l'indique le tableau suivant.

Table d'encodage

Caractère	**Code Morse**	**ICar$_{dec}$**	**OCar1**	**OCar2**	**OCar3**
"A"	● ▬	65	"00"	"01"	"11"
"D"	▬ ● ●	69	"01"	"00"	"00"
"G"	▬ ▬ ●	72	"01"	"01"	"00"
"I"	● ●	73	"00"	"00"	"11"
"K"	▬ ● ▬	75	"01"	"00"	"01"
"M"	▬ ▬	76	"01"	"01"	"11"
autres	N/A	autres	"11"	"11"	"11"

En sachant, que les caractères du code sont encodés comme suit :

- ● : "00"
- ▬ : "01"
- Rien : "11"

Donnez la description VHDL de l'encodeur de code Morse (librairies, entité et architecture).

Solution

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

entity code_morse is

   port (ICar : in std_logic_vector(6 downto 0);
         OCar1 : out std_logic_vector(1 downto 0);
         OCar2 : out std_logic_vector(1 downto 0);
         OCar3 : out std_logic_vector(1 downto 0));

end code_morse;

architecture dataflow of code_morse is
   type array_t is array (natural range<>) of std_logic_vector(5 downto 0);
   constant codes : array_t := ("000111", "010000", "010100", "000011", "010001", "010111",
"111111");


begin  -- dataflow
   process (ICar)
      variable k : integer;
   begin  -- process
      case to_integer(unsigned(ICar)) is
         when 65 => k := 0;            -- 'A'
         when 69 => k := 1;            -- 'D'
         when 72 => k := 2;            -- 'G'
         when 73 => k := 3;            -- 'I'
         when 75 =>  k := 4;           -- 'K'
         when 76 =>   k := 5;  -- 'M'
         when others => k := 6;
      end case;

      OCar1 <= codes(k)(5 downto 4);
      OCar2<= codes(k)(3 downto 2);
      OCar3 <= codes(k)(1 downto 0);
   end process;
end dataflow;
```

Question 2. (2 points)

Écrivez le code VHDL de l'architecture d'un filtre d'image dont l'entité « filtre_pixel » est présentée ci-après. Ce module reçoit en entrée le pixel à filtrer qui est un nombre non-signé et dont la taille est paramétrée. L'opération de filtrage s'effectue comme suit :

- si la valeur du pixel entrant est inférieure à Seuil1 alors pixel_out = Seuil1;
- si la valeur du pixel entrant est supérieure à Seuil2 alors pixel_out = Seuil2;
- sinon par défaut pixel_out = pixel_in.

En sachant que Seuil1 et Seuil2 sont des paramètres du module qui ont une valeur par défaut égale à 32 et 192 respectivement.

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

entity filtre_pixel is
    generic(
        N : positive := 8;              -- Nombre de bits des pixels
        Seuil1 : positive := 32;        -- le Seuil inférieur
        Seuil2 : positive := 192        -- le Seuil supérieur
    );
    port (
        pixel_in : in unsigned(N-1 downto 0);          -- Pixel entrant
        pixel_out : out unsigned(N-1 downto 0)         -- Pixel calcule
    );
end filtre_pixel;
```

Solution

```
entity filtre_pixel is
        generic(
                N : positive := 8;
                Seuil1 : positive := 32;
                Seuil2 : positive := 192
        );
        port (
                pixel_in : in unsigned(N-1 downto 0);
                pixel_out : out unsigned(N-1 downto 0)
        );
end filtre_pixel;

architecture behavior of filtre_pixel is

begin  -- behavior
   process (pixel_in)

   begin  -- process
        if pixel_in < Seuil1 then
           pixel_out <= to_unsigned(Seuil1, N);
        elsif pixel_in > Seuil2 then
           pixel_out <= to_unsigned(Seuil2, N);
        else
           pixel_out <= pixel_in;
        end if;
   end process;
end behavior;
```

Question 3. (8 points)

Écrivez le code VHDL d'un banc d'essai qui testera le filtre d'images décrit à la question 2. Les pixels entrants du filtre à tester auront une taille de 4 bits et les seuils : Seuil1 et Seuil2 seront égaux à 2 et 13 respectivement.

Le banc d'essai doit ouvrir un fichier d'images qui se nomme « image_in.pgm » qui a le format suivant :

Format_image ↵
nb_lignes␣nb_colonnes ↵
pixel_max_value ↵
$Pixel_0$␣$Pixel_1$␣$Pixel_2$␣…….␣$Pixel_{(\ nb\_lignes\ x\ nb\_colonnes-1\ ))}$ ↵

L'exemple suivant montre le contenu d'un fichier d'image dont le format est P5. Cette image comporte 3 lignes et trois colonnes en sachant que la valeur maximale des pixels devrait être égale à 15.

P5 ↵
3 3 ↵
15 ↵
3 2 0 3 12 14 17 8 9(la ligne de pixels) ↵

Les pixels sortant seront écrits dans un fichier de sortie qui se nommera « image_out.pgm ». Le format du fichier de sortie sera le même que celui du fichier d'entrée. Donc il faudra copier l'entête de l'image entrante dans le fichier de sortie avant d'écrire les pixels sortant.

Le nombre de pixels à traiter devra être égal à la taille de l'image (nb_lignes * nb_colonnes). Les pixels sont écrits sur une ligne unique dans le fichier d'image.

Le banc d'essai transmettra un nouveau pixel au module à tester à chaque 10 ns.

Durant la lecture de l'image entrante, si un pixel lu a une valeur supérieure à la valeur maximale précisée dans le fichier (pixel_max_value), le banc d'essai devra afficher un message d'erreur et arrêter la simulation.

Solution

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

entity filtre_pixel_tb is
   generic (Nbits : integer := 4);
end filtre_pixel_tb;

architecture beh of filtre_pixel_tb is
   signal pixel_in1 : unsigned(3 downto 0);
   signal pixel_out1 : unsigned(3 downto 0);

   constant filename_in : string := "image_in.pgm";
```

```
    constant filename_out : string := "image_out.pgm";
    file vecteurs : text open read_mode is filename_in;
    file resultats : text open write_mode is filename_out;

    component filtre_pixel
        generic(
            N : positive;
            Seuil1 : positive := 32;
            Seuil2 : positive := 192
        );
        port (
            pixel_in : in unsigned(N-1 downto 0);
            pixel_out : out unsigned(N-1 downto 0)
        );

    end component;

begin  -- beh

    UUT : filtre_pixel
        generic map (N => Nbits, Seuil1 => 2, Seuil2 => 13)
        port map (pixel_in => pixel_in1, pixel_out =>pixel_out1);

    process
    -- purpose:
        variable tampon_in : line; -- pointeur vers un objet de type string
        variable tampon_out : line; -- pointeur vers un objet de type string
        variable n : integer;
        variable c : character;
        variable nb_lignes : integer;
        variable nb_colonnes : integer;
        variable pixel_max_value : integer;
        variable nb_pixels_lus : integer := 0;

    begin  -- process
        -- lecture premiere ligne du fichier
        if not endfile(vecteurs) then
           readline(vecteurs, tampon_in);
           writeline(resultats, tampon_in);
        end if;

        -- lecture seconde ligne
        if not endfile(vecteurs) then
           readline(vecteurs, tampon_in);
           write(tampon_out, tampon_in.all);
           writeline(resultats, tampon_out);
           read(tampon_in, nb_lignes);
           read(tampon_in, c);
           read(tampon_in, nb_colonnes);
        end if;
```

```
        -- lecture troisieme ligne
        if not endfile(vecteurs) then
           readline(vecteurs, tampon_in);
           write(tampon_out, tampon_in.all);
           writeline(resultats, tampon_out);
           read(tampon_in, pixel_max_value);
        end if;

        -- lecture des pixels
        if not endfile(vecteurs) then
           readline(vecteurs, tampon_in);
        end if;
        while nb_pixels_lus < (nb_lignes * nb_colonnes)  loop
                read(tampon_in, n);
                read(tampon_in, c);
                -- verifier que la valuer du pixel lu est inferieure a pixel_max_value
                assert n < pixel_max_value
                   report "ERREUR : un pixel dont la valeur depasse la valeur maximale a ete lu
- ARRET DE LA SIMULATION" severity failure;

                -- envoi du pixel vers le UUT
                pixel_in1 <= to_unsigned(n, Nbits);
                nb_pixels_lus :=  nb_pixels_lus + 1;
                wait for 10 ns;

                -- ecriture du resultat dans le fichier de sortie
                write(tampon_out, integer'image(to_integer(pixel_out1)) & string'(" "));
           end loop;

           writeline(resultats, tampon_out);
        end if;

        deallocate(tampon_in); -- relâcher la mémoire du tampon
        deallocate(tampon_out); -- relâcher la mémoire du tampon
        report "simulation terminée" severity failure;

   end process;

end beh;
```

Question 4. (6 points)

Concevez un gestionnaire de boîte vocale. L'entité du gestionnaire est décrite comme suit :

```
entity gest_boite_vocale is
   port (
         reset_n : in std_logic;                 -- le reset du circuit
         clk : in std_logic;                     -- l'horloge
         tonalite : in std_logic;                -- la tonalite
         fin_lecture_accueil : in std_logic;     -- il est actif lorsque le message d'accueil est complete
         boite_pleine : in std_logic;            -- il signale que la  memoire de la boite vocale est pleine
         lecture_accueil : out std_logic;        -- il active le debut de la lecture du message d'accueil
         enregistrement : out std_logic          -- il active le debut du commencement d'un enregistrement
   );
end gest_boite_vocale;
```

Le gestionnaire fonctionne comme suit :

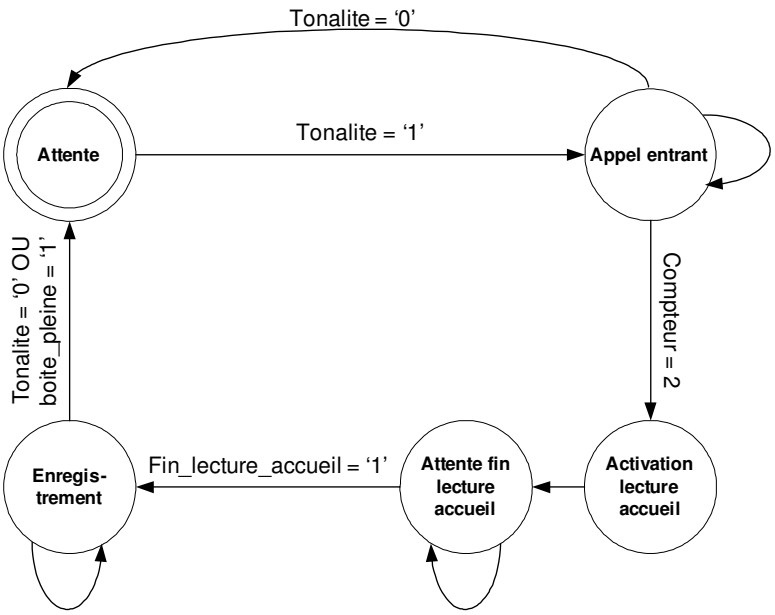
- Lorsque la tonalité est basse cela signifie qu'il n'y a aucun appel entrant. Dès que la tonalité est haute cela signifie qu'un appel entrant est actif.
- Si la tonalité reste haute pendant trois coups d'horloge consécutifs, alors le gestionnaire devra activer le port « lecture_accueil » pendant jusqu'à ce que le port « find_lecture_accueil » soit mis à '1'.
- Si avant les trois coups d'horloge la tonalité descend à '0' alors le gestionnaire se remettra en mode d'attente d'un nouvel appel.
- Après avoir activé la lecture du message d'accueil, le gestionnaire attendra que le port entrant fin_lecture_accueil soit activé ('1').
- Puis, le gestionnaire activera le port « enregistrement » à '1'.
- Il attendra que le port « tonalite » soit désactivé ('0') ou que le port « boite_pleine » soit activé pour arrêter l'enregistrement (mettre le port « enregistrement » à '0') et se remettre en mode attente d'un nouvel appel.
- Lorsque le circuit est réinitialisé (reset_n = '0'), toutes les sorties doivent être mises à 0et la machine à états du circuit doit être réinitialisée.

a) Donnez le diagramme d'états du gestionnaire de boîte vocale

b) Complétez le code VHDL de la machine à état correspondante en donnant l'architecture associée à l'entité décrite ci-avant.

Solution

a) Diagramme d'états



b) Code VHDL de l'architecture

```
architecture beh of gest_boite_vocale is

   type state_t is (ATTENTE, APPEL_ENTRANT, ACTIVATION_LECTURE_MACCUEIL,
ATTENTE_FIN_LECTURE_MACCUEIL, ENREGISTREMENT_MESSAGE);
   signal etat_courant : state_t := ATTENTE;
   signal etat_prochain : state_t := ATTENTE;
   signal compteur_courant : integer range 2 downto 0;
   signal reset_compteur : std_logic;
begin  -- beh
   -- mise a jour de l'etat courant ou des signaux registres
   process (clk, reset_n)

   begin  -- process
     -- activities triggered by asynchronous reset (active low)
     if reset_n = '0' then
        etat_courant <= ATTENTE;
     -- activities triggered by rising edge of clock
     elsif clk'event and clk = '1' then
        etat_courant <= etat_prochain;
     end if;
   end process;

   -- mise a jour de l'etat courant ou des signaux registres
   process (clk, reset_n)

   begin  -- process
     -- activities triggered by asynchronous reset (active low)
```

```
        if reset_n = '0' then
          compteur_courant <= 0;
        -- activities triggered by rising edge of clock
        elsif clk'event and clk = '1' then
          if reset_compteur = '0' and compteur_courant < 2 then
                compteur_courant <= compteur_courant + 1;
          else
                compteur_courant <= 0;
          end if;
        end if;
      end process;

      -- mise a jour de l'etat prochain
      process (etat_courant, tonalite, boite_pleine, compteur_courant)

      begin  -- process
        etat_prochain <= etat_courant;
        reset_compteur <= '0';
        case etat_courant is
          when ATTENTE =>
                if tonalite = '1' then
                  etat_prochain <= APPEL_ENTRANT;
                  reset_compteur <= '1';
                end if;
          when APPEL_ENTRANT  =>
                if compteur_courant = 2 then
                  etat_prochain <= ACTIVATION_LECTURE_MACCUEIL;
                elsif tonalite = '0' then
                  etat_prochain <= ATTENTE;
                end if;
          when ACTIVATION_LECTURE_MACCUEIL  =>
                etat_prochain <= ATTENTE_FIN_LECTURE_MACCUEIL;
          when ATTENTE_FIN_LECTURE_MACCUEIL  =>
                if fin_lecture_accueil = '1' then
                  etat_prochain <= ENREGISTREMENT_MESSAGE;
                end if;
          when ENREGISTREMENT_MESSAGE  =>
                if boite_pleine = '1' or tonalite = '0' then
                  etat_prochain <= ATTENTE;
                end if;
        end case;
      end process;


    --   -- mise a jour des sorties -- MOORE
      process (etat_courant)

      begin  -- process

        case etat_courant is
          when ACTIVATION_LECTURE_MACCUEIL  =>
```

```
                lecture_accueil <= '1';
                enregistrement <= '0';
            when ATTENTE_FIN_LECTURE_MACCUEIL  =>
                lecture_accueil <= '0';
                enregistrement <= '0';
            when ENREGISTREMENT_MESSAGE =>
                lecture_accueil <= '0';
                enregistrement <= '1';
            when others =>
                lecture_accueil <= '0';
                enregistrement <= '0';
        end case;
    end process;

    -- mise a jour des sorties -- MEALY
--    process (etat_courant, reset_n, tonalite, boite_pleine)

--    begin  -- process

--    case etat_courant is
--        when ACTIVATION_LECTURE_MACCUEIL  =>
--                lecture_accueil <= '1';
--                enregistrement <= '0';
--        when ATTENTE_FIN_LECTURE_MACCUEIL  =>
--                lecture_accueil <= '0';
--                enregistrement <= '0';
--        when ENREGISTREMENT_MESSAGE =>
--                if boite_pleine = '0' and tonalite = '1' then
--                    lecture_accueil  <= '0';
--                    enregistrement <= '1';
--                else
--                    lecture_accueil <= '0';
--                    enregistrement <= '0';
--                end if;
--        when others =>
--                lecture_accueil <= '0';
--                enregistrement <= '0';
--    end case;
--    if reset_n = '0' then
--        lecture_accueil <= '0';
--        enregistrement <= '0';
--    end if;
--    end process;

end beh;
```