Règles de déroulement de l'épreuve

Il n'est pas interdit de communiquer verbalement entre binômes tant que cela ne perturbe pas le déroulement correct de cette séance.
Il vous est absolument interdit, par contre, de vous déplacer, d'échanger des feuilles de brouillons, des notes ainsi que des fichiers.

Les fichiers nécessaires à la réalisation complète de cet examen médian sont comme d'habitude disponibles dans le répertoire median2014 ou median_OLD du fichier ressource2013.zip disponible sur le site personnel de Serge Moutou

I) Introduction

Nous allons partir d'une version corrigée du compteur de passages du TP6 et en faire un certain nombre de modifications. Dans cette épreuve , on ne vous demande pas de vous poser des questions sur l'utilité ou non des modifications mais seulement de les exécuter et de montrer que le travail résultant fonctionne toujours. Le but de l'épreuve est ainsi d'évaluer comment vous appréhendez des modifications à partir d'un schéma complexe décrit en VHDL ou en schématique.
Vous disposez d'une feuille réponse sur laquelle vous répondez et sur laquelle l'enseignant notera ce qu'il a vu fonctionner et sinon, quelques indications sur l'état d'avancement de votre travail.

L'objectif de cette épreuve est de réaliser une Modulation de Largeur d'Impulsion (MLI) dont la consigne est réalisée à l'aide du bouton tournant de la carte. Cette MLI sera visualisée sur une LED qui éclairera plus ou moins fort selon la consigne. Ceci sera réalisé en plusieurs étapes qui commencent par le compteur de passages. Voir Wikipédia :
http://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion

II) Compteur de passages corrigé.

Vous disposez du code source complet du compteur de passages (fichier compteur_passages.vhd dans les ressources). Un schéma partiel de cette version corrigée est sur la feuille réponse.

Travail à réaliser (3 pts)

1°) Compléter le schéma partiel sur la feuille réponse pour le composant transcod7segs du dessin. (On a ajouté les port map correspondants en bas du schéma)

2°) Réalisation matérielle :
On vous demande
- de prendre le fichier source (en annexe plus loin) ou disponible sur internet (fichier compteur_passages.vhd dans les ressources)
- de le couper en deux (là où cela est indiqué)
- de construire le fichier ucf
- de réaliser un projet comprenant les trois fichiers (2 VHD + 1 UCF)
- de compiler en corrigeant les éventuelles petites erreurs, d'essayer l'ensemble et de faire valider.

On rappelle que les tests se font à fréquence lente sur les deux interrupteurs "gauche" et "droite".
Le poids faible sera obligatoirement à droite.

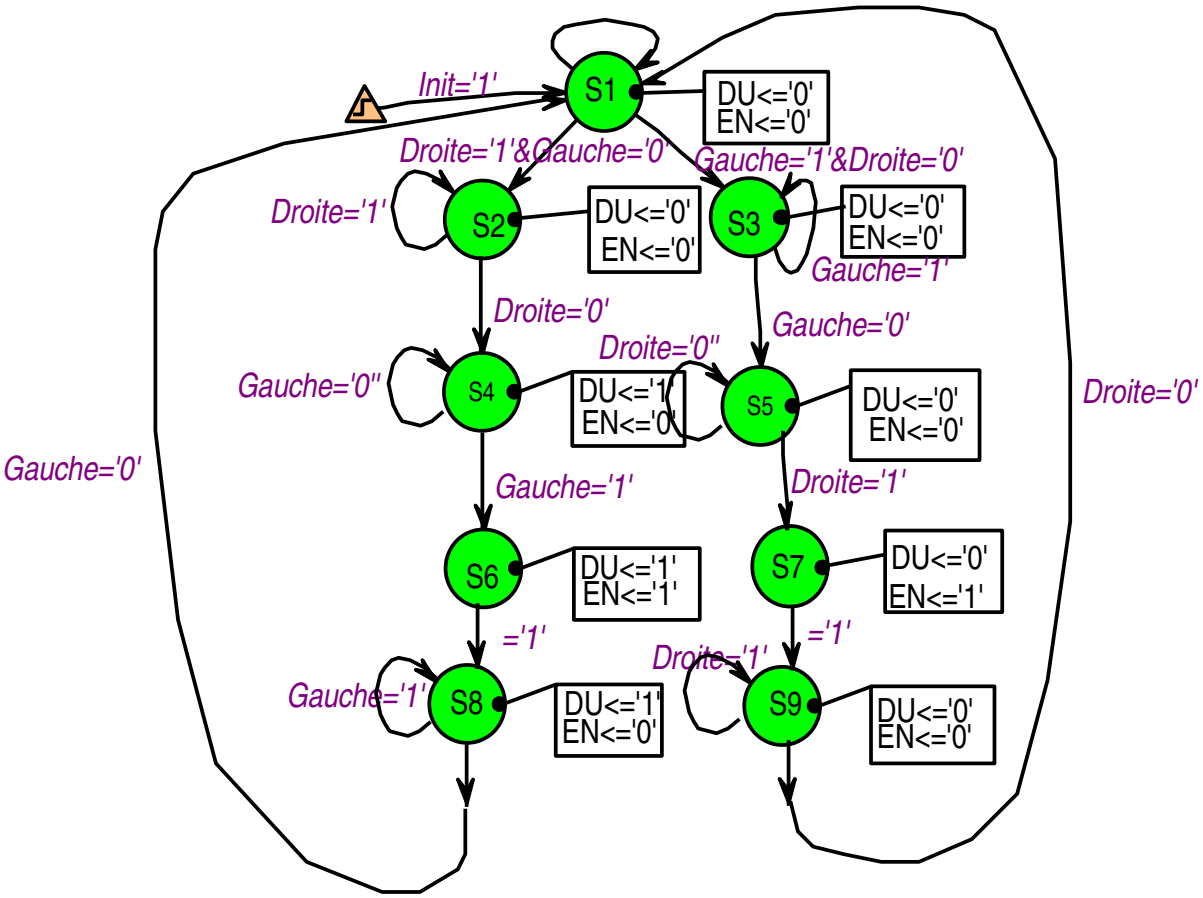
III) Modification du séquenceur

Pour réaliser notre modulation MLI on a besoin d'un séquenceur sur le compteur de passage un peu plus sophistiqué.

Travail à réaliser (5 pts)

1°) Dessiner sur votre feuille réponse le séquenceur de la correction sous forme de graphe d'évolution (états et transitions).

2°) Modifier le code du séquenceur correspondant au graphe d'évolution ci-après



IV) Utilisation d'un codeur incrémental

On trouve chez Xilinx un exemple de gestion du codeur incrémental. Celui-ci déplace une led allumée parmi les 8 leds éteintes et une led éteinte parmi les huit allumées si l'on appuie sur le codeur.

Travail à réaliser (2 points)

1°) On vous demande de compiler cet exemple et de le faire fonctionner. Tout est fourni, fichier VHDL et fichier UCF (left_right_leds.vhd et left_right_leds.ucf dans les ressources).

2°) Il est possible d'utiliser directement cet exemple pour commander le compteur de passages. Si on tourne dans un sens on incrémente le compteur, si on tourne dans l'autre sens on décrémente. On ne vous demande pas de le réaliser. Par contre, pouvez-vous citer sur votre feuille réponse quels sont les signaux (fils internes) qui pourraient jouer le rôle de EN et UD en lieu et place du séquenceur dans le fichier left_right_leds.vhd.

Soyez curieux et lisez attentivement le fichier UCF. La façon de déclarer les entrées correspondantes du décodeur comme PULLUP ou PULLDOWN est très importante sous peine de non fonctionnement. **<u>On l'utilisera dans la question suivante</u>**

V) Réalisation de la consigne à l'aide du bouton tournant

A partir de maintenant, nous allons procéder à la réalisation de la consigne (de notre MLI) en modifiant profondément le compteur de passages. Rechargez donc le projet correspondant en abandonnant le décodeur de la question IV).

En effet la réalisation d'une horloge lente comme nous l'avons réalisé en TD/TP est déconseillée dans un FPGA. Nous rappelons que nous avons procédé comme cela pour éviter les rebonds des interrupteurs. Les rebonds existent toujours. L'idée générale est donc d'utiliser l'horloge à 50 MHz pour tout le séquentiel mais d'ajouter après les entrées droite et gauche un anti-rebond.

Travail à réaliser (5 points)

La partie de code VHDL de l'anti-rebond vous est fournie dans le fichier "debounce.vhd".

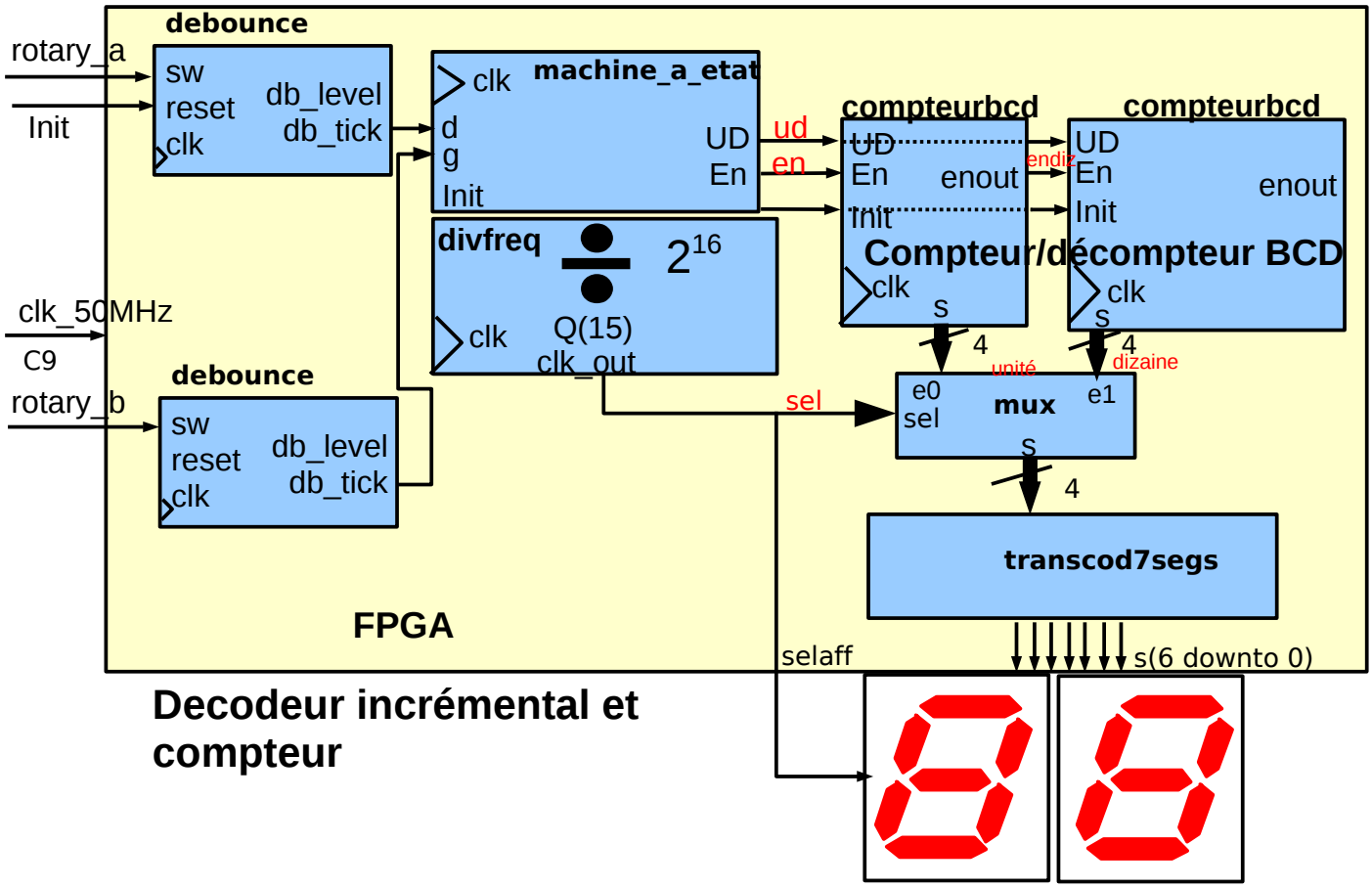
1°) Modifier le compteur "divfreq" pour qu'il ne sorte plus que la sortie "clock_out2" (qui sera relié à Q(15)) et éventuellement renommée "clock_out". On retirera donc "clock_out1" de l'entité, on redimensionnera le compteur en conséquence (sur 16 bits), et on modifiera les PORT MAP...

2°) Relier maintenant l'horloge de votre séquenceur, de vos compteurs BCD à l'horloge générale de 50 MHz

3°) Ajouter un anti-rebond sur chacune des entrées droite et gauche (figure ci-dessous)

4°) Modifier le fichier UCF et tester

Nous vous donnons la version du schématique du travail à réaliser.



Remarquez l'horloge commune à tous les circuits synchrone n'est pas dessinée, mais toutes les entrées clk sont reliées à"clk_50MHz". Le reset du deuxième "debounce" est aussi relié à init.

VI) Un compteur BCD cascadé sur deux digits et le comparateur associé

La MLI (PWM en anglais) est réalisée à l'aide d'un compteur BCD sur deux digits qui compte sans arrêt. Ces

deux compteurs seront comparés aux deux compteurs que l'on à réalisé en question V qui joueront donc le rôle de consigne.

Travail à réaliser (2 points)

Ajouter deux compteurs BDC (aux deux existants) en utilisant les compteurs BCD fournis dans la correction. (donc deux port map seulement) l'horloge utilisée encore l'horloge 50 MHz.

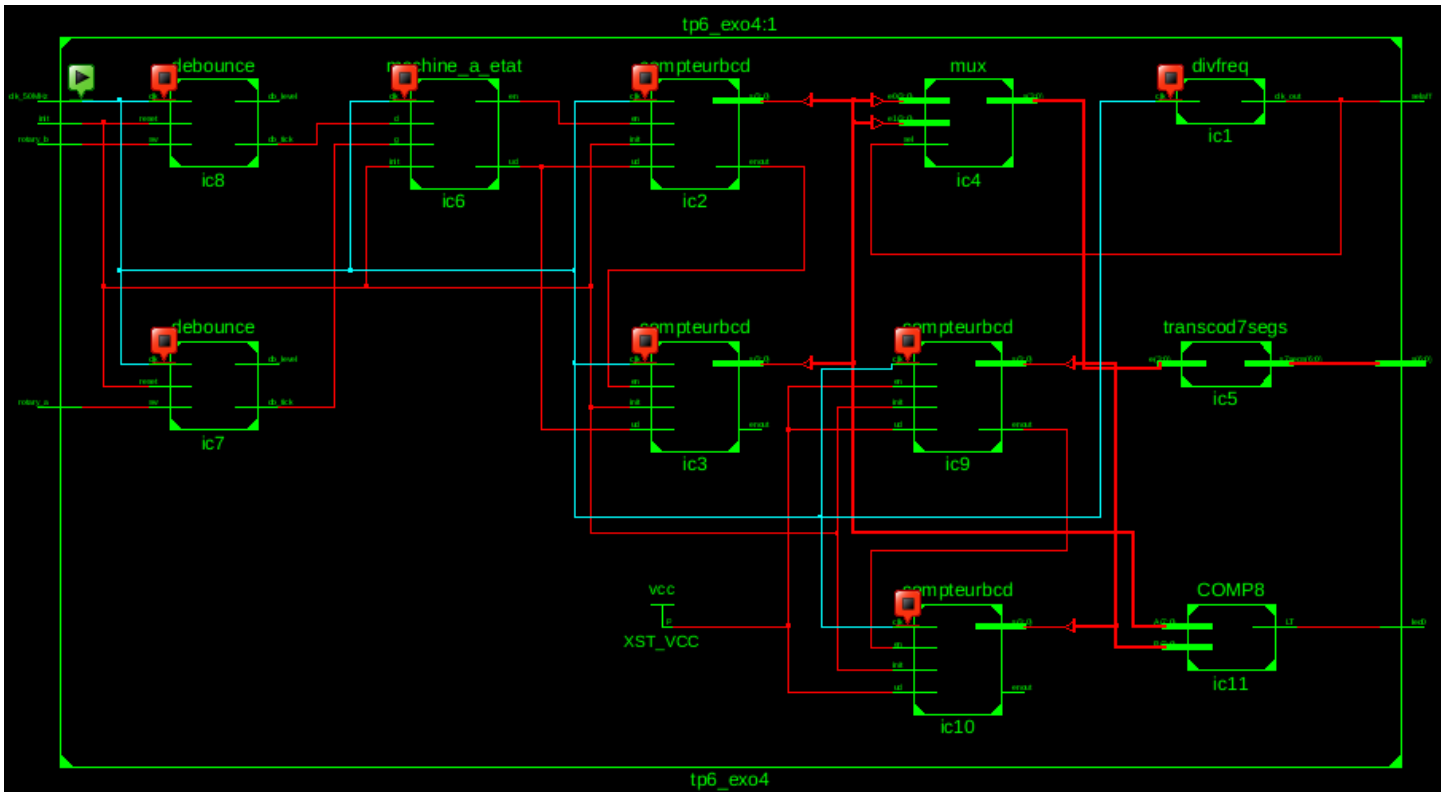
Travail à réaliser (4 points)

1°) Écrire un composant combinatoire complet capable de comparer deux valeurs de 8 bits dont la sortie LT est à 1 si A <B. On vous donne sa déclaration de composant :

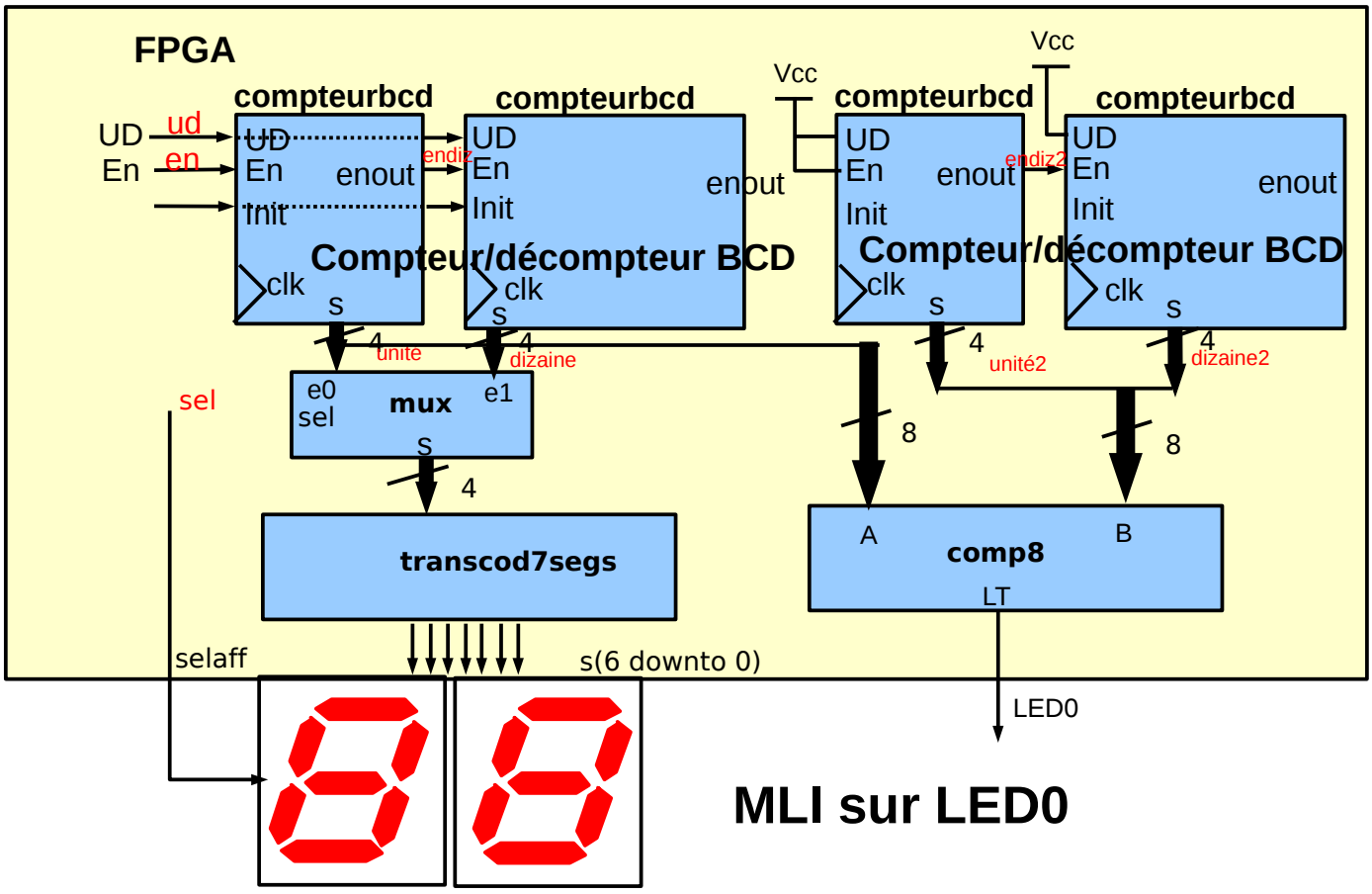
```
component COMP8
  port ( A  : in    std_logic_vector (7 downto 0);
         B  : in    std_logic_vector (7 downto 0);
         LT : out   std_logic);
end component;
```

Écrire l'entité et l'architecture correspondante. Insérer ce composant comme indiqué sur la figure ci-dessous. Tester votre MLI sur une LED. Son éclairement doit dépendre de la valeur affichée sur vos deux digits.

Voici une vue globale de ce que l'on doit obtenir :



Et maintenant un agrandissement de ce qu'il y a à faire dans cette question, la partie droite du schéma global :



Vous reconnaissez les deux compteurs BCD ajoutés complètement à droite, ainsi que le comparateur

2°) Il s'agit d'un comparateur binaire. Justifiez sur votre feuille réponse le fait que l'on peut l'utiliser en BCD.

VII) Réalisation complète en schématique pour finir

On vous demande dans cette partie de transformer tous les composants internes en symboles de schématique. Le plus facile pour faire cela est de réaliser un fichier VHDL par composant interne (copier coller du VHDL initial), de les ajouter tous dans le projet et de sélectionner un fichier VHD puis

Dans la fenêtre Design (tout en haut) : Design Utilities -> Create Schematic Symbol

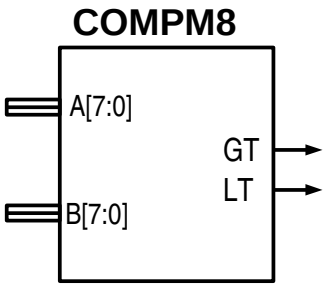
Vérifier alors la présence de vos fichiers SYM correspondants, c'est eux qui seront utilisés en schématique.

Nous utiliserons un comparateur tout fait de la librairie schématique. Il est donc inutile de transformer votre comparateur de la question VI en symbole.

Travail à réaliser (5 points)

Une fois ce travail réalisé, on vous demande d'assembler l'ensemble des composants bleus en un composant jaune (les couleurs sont celles du schéma) en utilisant l'outil de schématique. Arrangez-vous pour nommer vos entrées et sorties pour garder votre fichier UCF de la question précédente.

Voici la documentation du comparateur COMPM8 que l'on utilisera en lieu et place de **COMP8**.



La difficulté de cet exercice est de câbler des bus de tailles différentes : deux bus de 4 fils avec un bus de 8 fils. Pour réaliser cela vous devez dans l'odre :

- tirer un fil sur le bus 8 bits et le nommer (avec l'extension [7:0])
- tirer un fil sur le bus 4 bits et le nommer avec le même nom en changeant les numéros.

N'oubliez pas que tous les fils portant même nom sont automatiquement reliés. Ils deviennent alors difficiles à effacer individuellement.

ANNEXE

```
--Version pour spartan 3E
-- Compteur de passage sur 2 digits
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity tp6_exo4 is port (
      gauche,droite: in std_logic;
      clk_50MHz : in std_logic;
      init : in std_logic;
      selaff : out std_logic; -- pour test sur carte spartan3E
      s: out std_logic_vector(6 downto 0)
      );
end entity;

architecture behavior of tp6_exo4 is
      component machine_a_etat is port (
            clk,init : in std_logic;
            g,d : in std_logic;
            en,ud : out std_logic
            );
      end component;
      component divfreq is port (
            clk : in std_logic;
            clk_out1,clk_out2 : out std_logic );
      end component;
      component compteur is port (
            clk :in std_logic;
            init : in std_logic;
            s: out std_logic_vector(7 downto 0)
            );
      end component;
      component compteurbcd is port (
            clk :in std_logic;
            en : in std_logic;
            ud: in std_logic;
            init : in std_logic;
            enout : out std_logic;
            s: out std_logic_vector(3 downto 0)
            );
      end component;
      component transcod7segs IS PORT(
     e : in std_logic_vector(3 downto 0);
     s7segs : out std_logic_vector(6 downto 0));
   END component;
      component mux is port (
            sel: in std_logic;
            e0,e1 : in std_logic_vector(3 downto 0);
            s : out std_logic_vector(3 downto 0)
      );
      end component;
      signal clk_lente,sel,endiz : std_logic;
      signal nb4bits,unite,dizaine : std_logic_vector(3 downto 0);
      signal nb : std_logic_vector(7 downto 0);
      signal en,ud :  std_logic;

begin
  ic1: divfreq port map ( clk =>clk_50MHz, clk_out1=> clk_lente,clk_out2=>sel);
```

```
  ic2: compteurbcd port map( clk => clk_lente, ud=>ud,en=>en,init => init,
s=>unite,enout=>endiz);
  ic3: compteurbcd port map( clk => clk_lente, ud=>ud,en=>endiz,init => init,
s=>dizaine);
  ic4: mux port map ( sel=>sel, e0=>unite, e1=>dizaine, s=>nb4bits);
  ic5: transcod7segs port map (e => nb4bits, s7segs=> s);
  ic6: machine_a_etat port map
(clk=>clk_lente,init=>init,g=>gauche,d=>droite,en=>en,ud=>ud);

  selaff <= sel; -- test spartan3E
end behavior;
---------------------------------- couper le fichier ici
-------------------------------------
-- description du graphe
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity machine_a_etat is port (
      clk,init : in std_logic;
      g,d : in std_logic;
      en,ud : out std_logic
      );
end machine_a_etat;

architecture archi_machine of machine_a_etat is
      type type_etat is (s1,s2,s3,s4,s5,s6,s7);
      signal next_etat,reg_etat:type_etat;
begin
      valide_etat:process(clk)
        begin
                if rising_edge(clk) then
                        if init='1' then
                                reg_etat<=S1;
                        else
                                reg_etat<=next_etat;
                        end if;
                end if;
        end process valide_etat;
        etat_suivant: process (reg_etat,d,g)
        begin
                --next_etat<=reg_etat;
                case reg_etat is
                        when S1=>
                                if d='1' and g='0' then
                                        next_etat<=S2;
                                elsif g='1' and d='0' then
                                        next_etat<=S3;
                                else
                                        next_etat<=S1;
                                end if;
                        when S2=>
                                if g='1' then
                                        next_etat<=S4;
                                else
                                        next_etat<=S2;
                                end if;
                        when S3=>
                                if d='1' then
                                        next_etat<=S5;
                                else
                                        next_etat<=s3;
                                end if;
```

```
                        when S4=>next_etat<=S6;

                        when S5=>next_etat<=S7;

                        when S6=>
                                if g='0' then
                                        next_etat<=S1;
                                else
                                        next_etat<=S6;
                                end if;
                        when S7=>
                                if d='0' then
                                        next_etat<=S1;
                                else
                                        next_etat<=S6;
                                end if;
                        end case;
        end process etat_suivant;
        -- gestion des actions
        process(reg_etat)
        begin
                if reg_etat = s2 then
                        ud <= '1';
                elsif reg_etat = s4 then
                        ud <= '1';
                elsif reg_etat = s6 then
                        ud <= '1';
                else
                        ud <= '0';
                end if;
        end process;
        process(reg_etat)
        begin
                if reg_etat = s4 then
                        en <= '1';
                elsif reg_etat = s5 then
                        en <= '1';
                else
                        en <= '0';
                end if;
        end process;
end archi_machine;


-- description de la memoire de transcodage
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mux is port (
        sel: in std_logic;
        e0,e1 : in std_logic_vector(3 downto 0);
        s : out std_logic_vector(3 downto 0)
        );
end entity;
architecture behavior of mux is
begin
        with sel select
                s <= e0 when '0',
                        e1 when others;
end behavior;

-- description de la memoire de transcodage
```

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
ENTITY transcod7segs IS PORT(
  e : in std_logic_vector(3 downto 0);
  s7segs : out std_logic_vector(6 downto 0));
END transcod7segs;
ARCHITECTURE arch of transcod7segs IS
BEGIN
  with e select
    s7segs <= "1111110" when "0000",
              "0110000" when "0001",
              "1101101" when "0010",
              "1111001" when "0011",
              "0110011" when "0100",
              "1011011" when "0101",
              "1011111" when "0110",
              "1110000" when "0111",
              "1111111" when "1000",
              "1111011" when "1001",
              "1110111" when "1010",
              "0011111" when "1011",
              "1001110" when "1100",
              "0111101" when "1101",
              "1001111" when "1110",
              "1000111" when others;
END;

-- description du premier composant : diviseur de frequence
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity divfreq is port (
            clk : in std_logic;
            clk_out1,clk_out2 : out std_logic );
end entity;

architecture behavior of divfreq is
signal  n : std_logic_vector(23 downto 0);
begin
      -- division de la frequence de base de 50MHz vers 3Hz
      divfreq :process(clk) begin
            if clk'event and clk='1' then
                  n <= n+1;
            end if;
      end process;
            clk_out1 <= n(23); -- visualisation de l'horloge
            clk_out2 <= n(17); -- gestion afficheurs 7 segments
end behavior;

-- description du composant compteur/decompteur cascadable
library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

      entity compteurbcd is port (
            clk :in std_logic;
```

```
            en : in std_logic;
            init : in std_logic;
            ud : in std_logic;
            enout : out std_logic;
            s: out std_logic_vector(3 downto 0)
            );
      end entity;

      architecture behavior of compteurbcd is
      signal n : std_logic_vector(3 downto 0);
      begin
      increment : process(clk) begin
            if clk'event and clk='1' then
                  if init ='1' then
                        n <= (others => '0');
                  elsif en='1' then
                        if ud = '1' then  --up
                              if n<9 then
                                    n <= n + 1 ;
                              else
                                    n <= (others => '0');
                              end if;
                        else -- down
                              if n=0 then
                                    n <= "1001";
                              elsif n<10 then
                                    n <= n - 1 ;
                              else
                                    n <= (others =>'0');
                              end if;
                        end if;
                  end if;
            end if;
      end process;
      enableout: process(n,en,ud)
      begin
            if en='1' then
                  if ud='1' and n=9 then
                        enout<= '1';
                  elsif ud='0' and n=0 then
                        enout<='1';
                  else
                        enout<='0';
                  end if;
            else -- ajouté 4/5/2011 pour eviter comptage intempestif sur dizaine
              enout<='0';
            end if;
      end process;
      s <= n;
end behavior;
```