INF3500 : Conception et réalisation de systèmes numériques

Corrigé de l'examen final

Vendredi 17 août 2012

Durée: 2h30.

Pondération: 40%.

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 dans le cahier fourni.
- Ne pas remettre le questionnaire.
- Ne posez pas de question durant l'examen. En cas de doute sur le sens d'une question, énoncez clairement toute supposition que vous faites.

Question 1. (10 points)

Considérez le code VHDL donné à l'annexe I, qui est celui du processeur à usage général qui vous avait été remis au laboratoire.

a) Proposez une modification à la ligne suivante:

```
(x"A700", x"0005", x"AC00", x"0003", x"137C", others => (others => '1'));
```

de sorte que le processeur exécute l'algorithme suivant:

```
acc = -45;
Pour i = 1 à 100, répéter
        Si ( i modulo 2 = 0 ) alors
                acc = acc + i;
        autrement
                acc = acc - i;
        Fin si
Fin répéter
```

Réponse :

```
 0 (x"A000", x"FFD3", -- charge -45 dans R0 (ACC)
 2 x"A100", x"0001", -- charge +1 dans R1 (I)
 4 x"A200", x"0065", -- charge +101 dans R2 (MAX_I+1)
 6 x"A300", x"0001", -- charge +1 (MASK/INC)dans R3
 8 x"1421", -- R4 ← R2-R1 (BUFFER)
 9 x"C110", -- JUMP TO #16 IF R4 IS ZERO
10 x"5413", -- R4 ← R1 AND R3
11 x"C20E", -- JUMP TO #14 IF R4 IS NOT ZERO
12 x"0001", -- R0 ← R0+R1 (ACC = ACC + I)
13 x"C00F", -- JUMP TO #15
14 x"1001", -- R0 ← R0-R1 (ACC = ACC - I)
15 x"0113", -- R1 ← R1+R3 (I = I + 1)
16 x"C008", -- JUMP TO #8
16 others => (others => '1'));
```

b) Quelle est la valeur de "acc" à la fin de l'exécution de l'algorithme? Justifiez brièvement.

Réponse :

À la fin de l'algorithme, acc = -45 + 2550 – 2500 = +5

c) Donnez une évaluation du nombre de cycles d'horloge nécessaires à l'exécution de votre implémentation de l'algorithme. Cet estimé ne doit pas être exact, mais il doit être suffisamment précis et adéquatement justifié.

Réponse :

Il aura 4 chargements immédiats, chacun nécessitant 4 cycles.
Les instructions #8 et 9 seront exécutée 101 fois, chacune nécessitant 3 cycles.
Les instructions #10 et #11 seront exécutées 100, 3 cycles chacune.
Les instructions #12 et #13 seront exécutées 50, 3 cycles chacune.
L'instruction #14 sera exécutée 50, 3 cycles à chaque fois.
Les instructions #15 et #16 seront exécutées 100, 3 cycles chacunes.

Total : 4x4 + 2x101x3 + 2x100x3 + 2x50x3 +50x3 + 2x100x3 = 2272 cycles.

Question 2. (8 points)

On vous demande de réaliser un circuit permettant de trouver la valeur maximale de cinq nombres signés donnés en entrée : a, b, c, d et e. Ainsi, si (a, b, c, d, e) = (+3, +8, +5, -7, +2), la sortie devrait être z = +8.

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

entity max5 is
generic(
    w : positive := 8
);
port(
    a : in signed(w-1 downto 0);
    b : in signed(w-1 downto 0);
    c : in signed(w-1 downto 0);
    d : in signed(w-1 downto 0);
    e : in signed(w-1 downto 0);
    z : out signed(w-1 downto 0)
);
end max5;
```

a) Écrivez le code VHDL synthétisable de l'entité **`max5`**.

Réponse :

```
architecture arch of max5 is

    signal max_ab : signed(w-1 downto 0);
    signal max_cd : signed(w-1 downto 0);
    signal max_abcd : signed(w-1 downto 0);
    signal max_abcde : signed(w-1 downto 0);

begin

    max_ab <= a when a > b else b;
    max_cd <= c when c > d else d;
    max_abcd <= max_ab when max_ab > max_cd else max_cd;
    max_abcde <= max_abcd when max_abcd > e else e;

    z <= max_abcde;

end arch;
```

b) Quelle est le nombre de vecteurs de test nécessaires pour effectuer un test exhaustif de l'entité **`max5`** ? Justifiez votre réponse par un calcul.

2^{5w}

On vous propose cinq classes pour chacune des entrées du circuit **`max5`** (on suppose w≥3) :

{ $\{-2^{w-1}, -2^{w-1} + 1, \ldots -2^{w-2} - 1\}$,
$\{-2^{w-2}, -2^{w-2} + 1, \ldots -2^{w-3} - 1\}$,
$\{-2^{w-3}, \ldots +2^{w-3} - 1\}$,
$\{+2^{w-3}, +2^{w-2}+1, \ldots +2^{w-2} - 1\}$,
$\{+2^{w-2}, +2^{w-2}+1, \ldots +2^{w-1} - 1\}$ }

c) Quelle est la taille minimale du vecteur de test pour effectuer un test faible sur l'entité **`max5`**, et ce en recourant aux classes identifiées précédemment. Justifiez votre réponse.

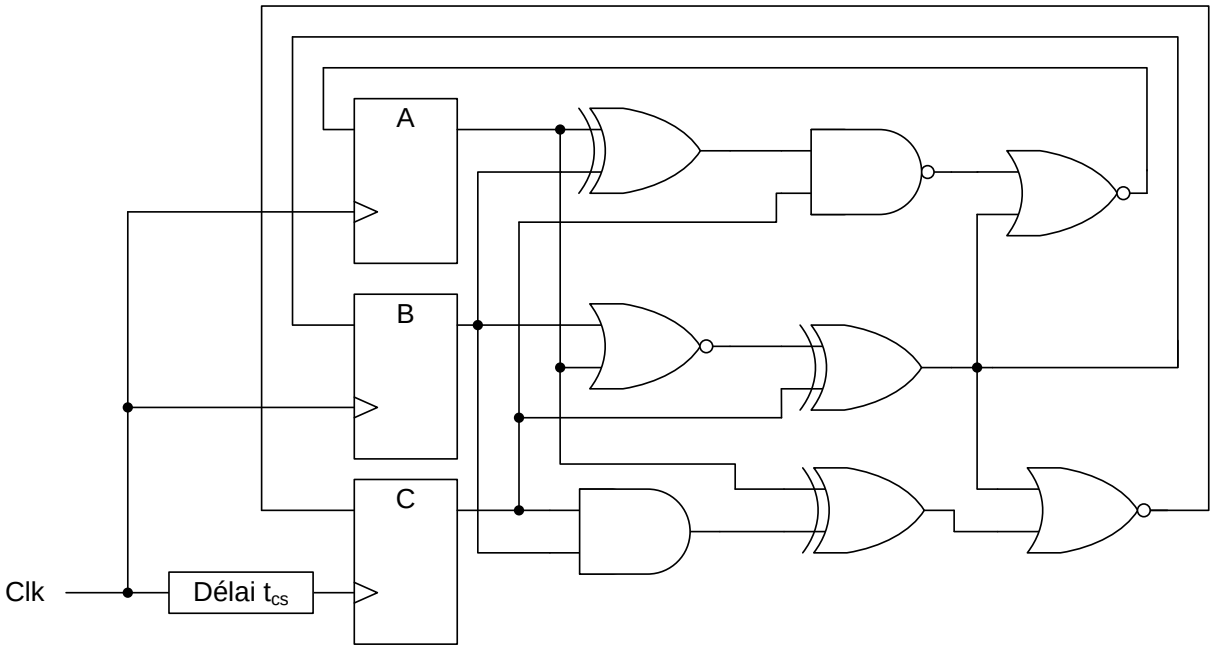
5 x 5 = 25, chaque entrée prenant une fois un exemplaire d'une se ses classes.

d) Quelle est la taille minimale du vecteur de test pour effectuer un test fort sur l'entité **`max5`**, et ce en recourant aux classes identifiées précédemment. Justifiez votre réponse.

5^5, chaque entrée prenant une fois un exemplaire d'une se ses classes, de sorte à couvrir toutes les combinaisons possibles.

Question 3. (10 points)

Considérez le circuit suivant. Les délais des portes logiques combinatoires sont respectivement de 4 ns pour un XOR, 3 ns pour un ET/OU, 2 ns pour un NON ET/NON OU. Les bascules ont un temps de préparation de 1 ns, un temps de maintien de 0.5 ns et un temps de propagation de 2 ns. On néglige le délai associé aux fils.



En supposant dans un premier temps que le délai de retard sur l'horloge est absent du circuit (t_{cs} = 0):

a) Donnez le chemin critique en indiquant les composantes sur ce chemin, et donnez la fréquence maximale d'horloge.

src/dest	A	B	C
A	8+3 ns	6+3 ns	6+3 ns
B	8+3 ns	6+3 ns	**9+3 ns**
C	4+3 ns	4+3 ns	**9+3 ns**

Le chemin critique part de B/C et finit à C. La fréquence maximale est 1/12ns = 83.33 MHz.

b) Donnez la marge libre de préparation pour les autres chemins.

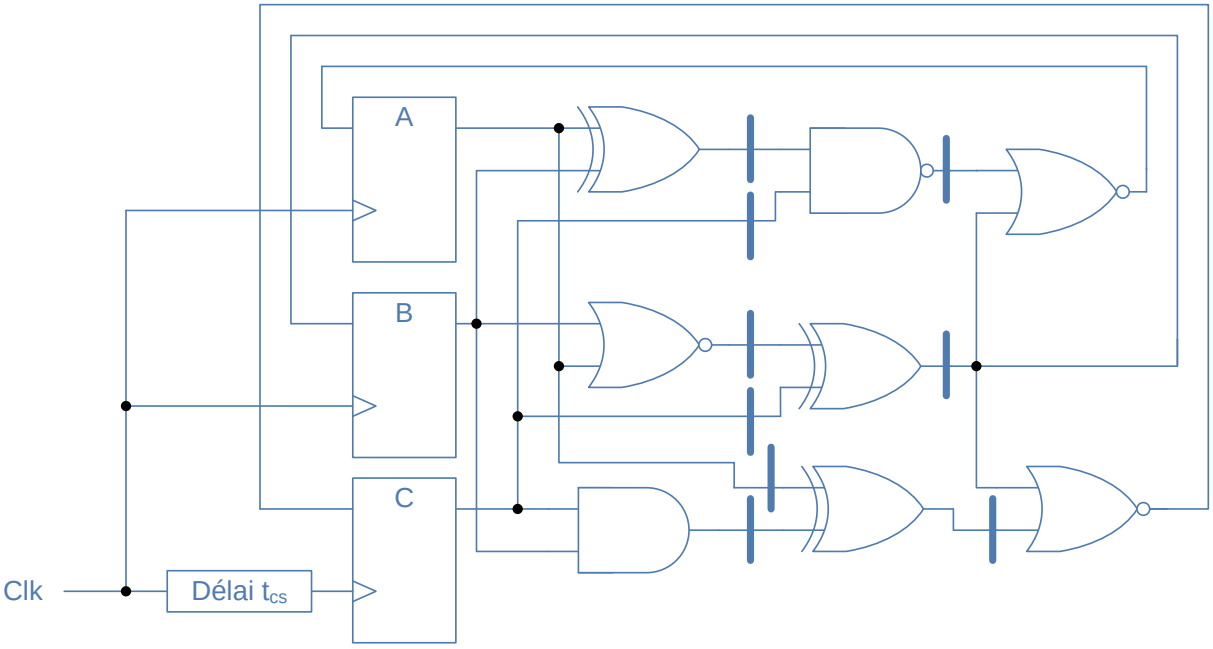
src/dest	A	B	C
A	1 ns	3 ns	3 ns
B	1 ns	3 ns	**0 ns**
C	5 ns	5 ns	**0 ns**

c) En supposant que vous pouvez ajouter autant de registres que vous le voulez, quelle est la fréquence maximale d'opération du circuit si on procédait à son pipelinage? Expliquez clairement votre réponse.

t_{comb} maximal = 4ns. Freq. Max = 1/(4+3) = 142.85 MHz.

d) Quel est le nombre minimal de registres à ajouter pour obtenir la fréquence maximale obtenue à la question (c) ?

Au total 9 :



En supposant que le délai de retard du signal d'horloge vers la bascule C est t_{cs}=1 ns:

e) Donnez le chemin critique en indiquant les composantes sur ce chemin, et donnez la fréquence maximale d'horloge.

src/dest	A	B	C
A	**8+3 ns**	6+3 ns	5+3 ns
B	**8+3 ns**	6+3 ns	**8+3 ns**
C	4+3 ns	4+3 ns	**8+3 ns**

Les chemins critiques partent de A/B à A ou de B/C à C. La fréquence maximale est 1/11ns = 90.9 MHz.

f) Donnez la marge libre de préparation pour les autres chemins.

src/dest	A	B	C
A	**0 ns**	2 ns	3 ns
B	**0 ns**	2 ns	**0 ns**
C	4 ns	4 ns	**0 ns**

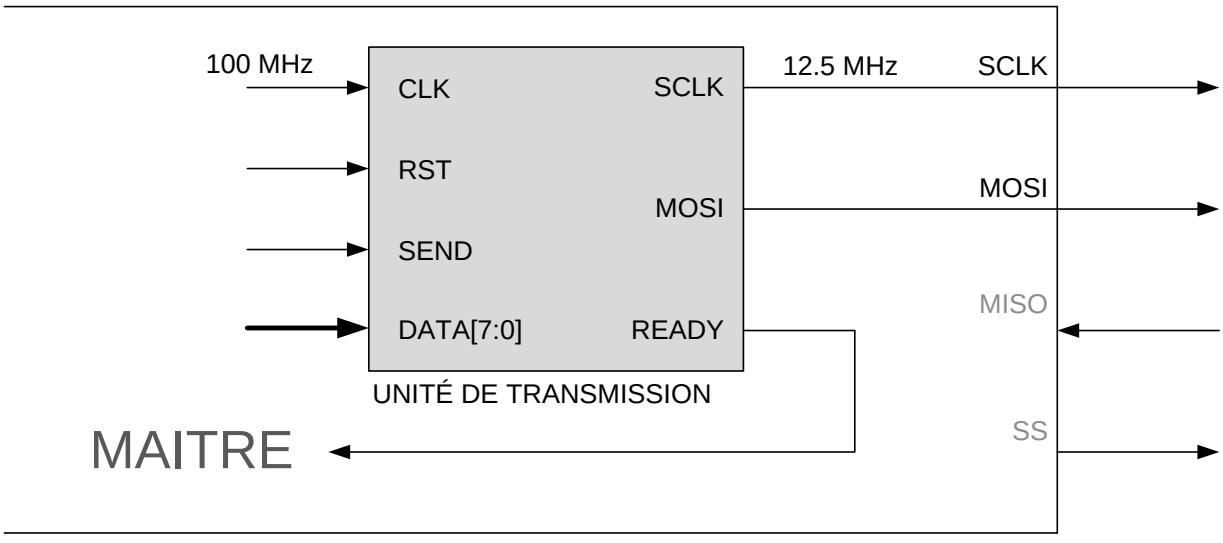
Question 4. (12 points)

On désire concevoir un lien SPI (*serial peripheral interface bus*). Une liaison SPI est un bus synchrone qui opère en full duplex, c'est-à-dire dans les deux sens et en même temps (c'est le cas également de la communication RS232 que vous avez manipulée au laboratoire). La liaison point à point dans un lien de communication SPI se fait entre un "maître" et un "esclave". La figure suivante illustre le lien de communication entre les deux unités:

Voici une description des signaux du lien de communication:

- ✓ SCLK : est un signal d'horloge généré par le maître durant la communication et par lequel les données (sur les lignes MOSI/MISO) sont synchronisées.
- ✓ MOSI : est un signal de données de 1 bit permettant au maître d'envoyer des données à l'esclave. L'acronyme est pour Master Out, Slave In.
- ✓ MISO : est un signal de données de 1 bit permettant à l'esclave d'envoyer des données au maître lorsque le maître initie une communication. L'acronyme signifie Master In, Slave Out.
- ✓ $\overline{SS}$: est un signal actif bas servant à sélectionner l'esclave avec lequel le maître veut communiquer. Il est utile dans une topologie où plusieurs esclaves sont attachés au maître. Nous ne le considérons pas ici.

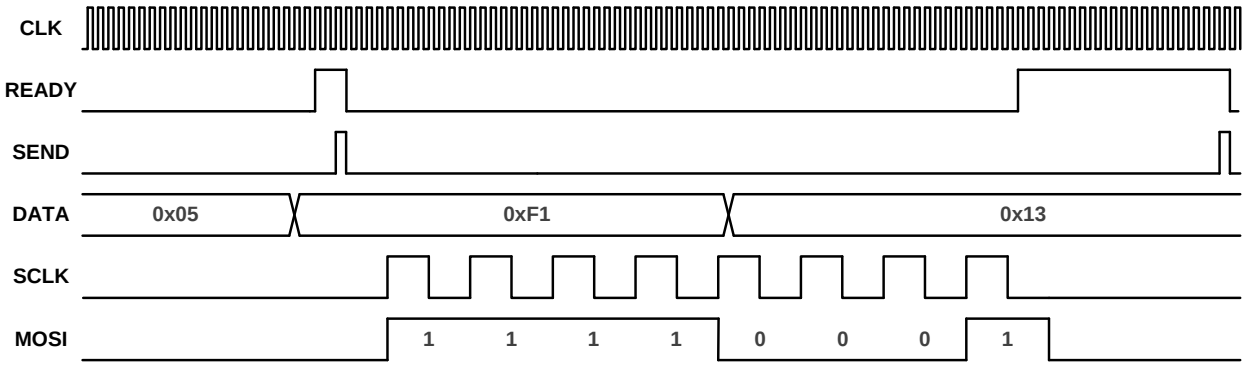
Nous désirons réaliser l'unité de transmission du maître permettant de générer le signal d'horloge et d'envoyer des données vers l'esclave (unité de transmission), tel qu'illustré à la figure suivante:



Voici une description des signaux du bloc à réaliser:

- ✓ CLK : signal d'horloge à 100 MHz.
- ✓ RST : signal de réinitialisation, synchrone sur CLK.
- ✓ SEND : signal indiquant au circuit d'envoyer les données présentent sur le bus DATA. Ce signal n'a aucun effet si READY vaut 0.
- ✓ DATA: bus de données à envoyer à travers le bus sériel MOSI.
- ✓ SCLK: signal d'horloge SPI, synchrone avec CLK et actif uniquement durant une communication et d'une fréquence de 12.5 MHz.
- ✓ MOSI : signal de donnée SPI permettant au maître d'envoyer DATA de façon sérielle vers l'esclave. DATA est traité de façon que les bits de poids supérieurs sont envoyés en premier,et les bits de poids inférieurs en dernier.
- ✓ READY: signal indiquant que le circuit est prêt à recevoir de nouvelles données à envoyer vers l'esclave.

La figure suivante illustre le fonctionnement de l'unité de transmission à réaliser. Dans cet exemple, on voit le maître envoyer le mot 0xF1 sur la ligne sérielle MOSI une fois que READY vaut 1. On voit également que l'horloge SCLK est activée uniquement durant la transmission (on voit donc 8 fronts montants).

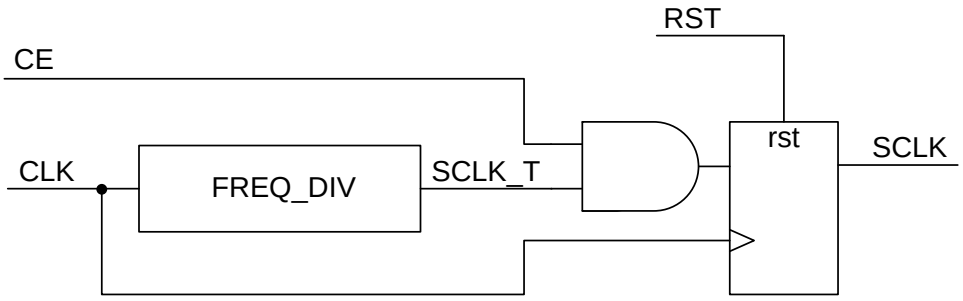


a) Donner en VHDL l'architecture du circuit diviseur de fréquence dont la déclaration d'entité est donnée ci-après. Le diviseur de fréquence prend en entrée un signal d'horloge de 100 MHz et produit en sortie une horloge de 12.5 MHz. Le circuit est insensible au signal de réinitialisation RST car on désire que le diviseur de fréquence soit indépendant de l'état du reste de l'unité de transmission.

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

entity freq_div is
port(
        clk100MHz  : in STD_LOGIC;
        clk12p5MHz : out STD_LOGIC
    );
end freq_div;
```

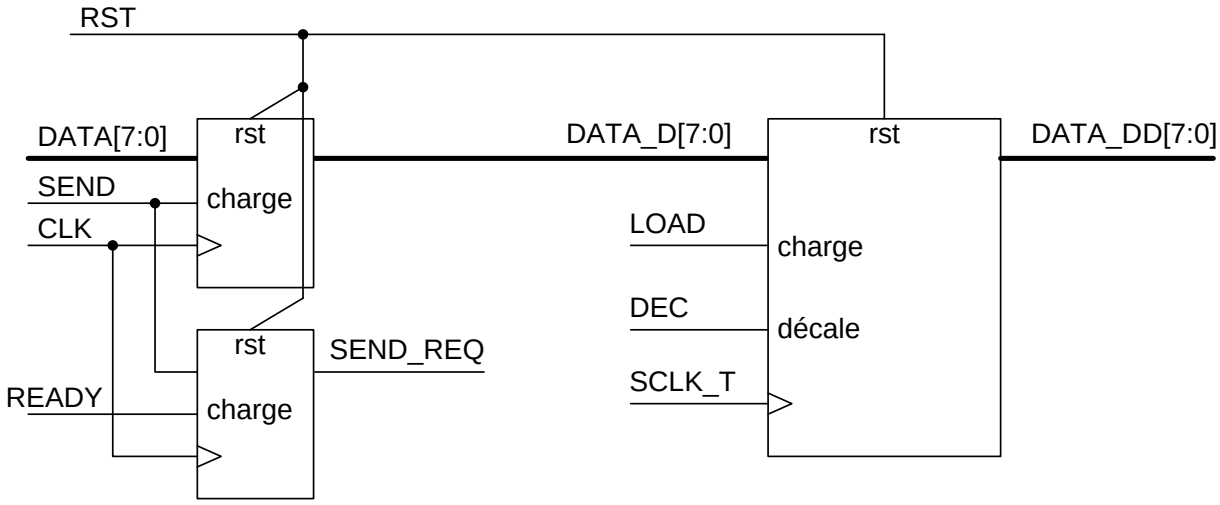
```
architecture clockgen of freq_div is
signal compteur : unsigned(2 downto 0) := (others => '0');
begin


    process (clk100MHz) is
    begin
        if rising_edge(clk100MHz) then
            compteur <= compteur + 1;
        end if;
    end process;

    clk12p5MHz <= compteur(compteur'left);

end clockgen;
```

b) On vous propose d'utiliser le circuit suivant pour générer le signal SCLK en exploitant le diviseur de fréquence de la question (a). Expliquez l'utilité d'une telle approche, autrement dit expliquez la raison d'être de la bascule de sortie.



L'horloge en sortie est stable puisqu'elle provient d'un registre et non d'un chemin combinatoire, qui serait susceptible aux aléas sur CE ou SCLK_T.

c) On vous propose de réaliser le circuit de la figure suivante, composé de registres synchronisés sur CLK et d'un registre à décalage synchronisé sur SCLK_T. Ce circuit nous sera utile dans la réalisation de l'**unité de transmission**. Donnez une description VHDL de ce bloc dont la déclaration d'entité (code VHDL) est donnée ci-après. Les spécifications de ce circuit sont énoncées à la page suivante.



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

entity decaleur is

port(
        RST         : in std_logic;
        CLK         : in std_logic;
        SCLK_T      : in std_logic;
        DEC         : in std_logic;
        LOAD        : in std_logic;
        READY       : in std_logic;
        SEND        : in std_logic;
        DATA        : in std_logic_vector(7 downto 0);
        DATA_DD     : out std_logic_vector(7 downto 0);
        SEND_REQ    : out std_logic
    );
end decaleur;
```

- ✓ Le signal de réinitialisation de toutes les registres est synchrone avec l'horloge et actif haut.
- ✓ L'entrée DATA est échantillonnée sur un front montant du signal CLK (100 MHz) lorsque SEND vaut 1.
- ✓ Le signal SEND est registré sur un front montant du signal CLK et donné en sortie sur le signal SEND_REQ.
- ✓ Le registre à décalage fonctionne sur l'horloge SCLK_T (12.5 MHz).
- ✓ Le signal LOAD a priorité sur le signal DEC et instruit le registre à décalage de charger la valeur en entrée (DATA_D)
- ✓ Le registre à déclage décale la valeur DATA_DD vers la gauche au front montant de l'horloge lorsque DEC = 1, à moins que LOAD = 1.

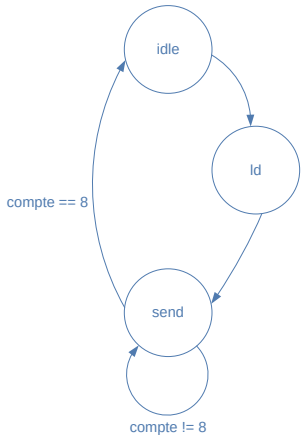
```
architecture arch of decaleur is
    signal s_data_d : std_logic_vector(7 downto 0) := (others => '0');
    signal s_data_dd : std_logic_vector(7 downto 0) := (others => '0');
    signal s_send_req : std_logic := '0';
begin
    process( CLK )is
    begin
        if ( rising_edge( CLK ) ) then
        if ( RST = '1' )then
            s_data_d <= (others => '0');
            s_send_req <= '0';
        else
            s_data_d <= DATA;
            if ( READY = '1' ) then
                s_send_req <= SEND;
            end if;
        end if;
        end if;
    end process;

    process( SCLK_T )is
    begin
        if ( rising_edge( SCLK_T ) ) then
        if ( RST = '1' )then
            s_data_dd <= (others => '0');
        else
            if ( LOAD = '1' ) then
```

```
                s_data_dd <= s_data_d;
            else
                if( DEC = '1' ) then
                    s_data_dd <= s_data_dd(w-2 downto 0) & '0';
                end if;
            end if;
        end if;
        end if;
    end process;

    SEND_REQ <= s_send_req;
    DATA_DD <= s_data_dd;

end arch;
```

On désire contrôler l'unité de transmission suivant l'horloge SCLK_T. Les entrées et sorties de cette unité de contrôle sont données dans la déclaration d'entité suivante. Le signal CE est celui du circuit de la question (b).

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

entity ctrl_transmission is
    port(
        RST         : in std_logic;
        SCLK_T      : in std_logic;
        SEND_REQ    : in std_logic
        READY       : out std_logic;
        LOAD        : out std_logic;
        DEC         : out std_logic;
        CE          : out std_logic
    );
end ctrl_transmission;
```

e) Dessinez le diagramme d'états de la machine à états utilisée par cette unité de contrôle.

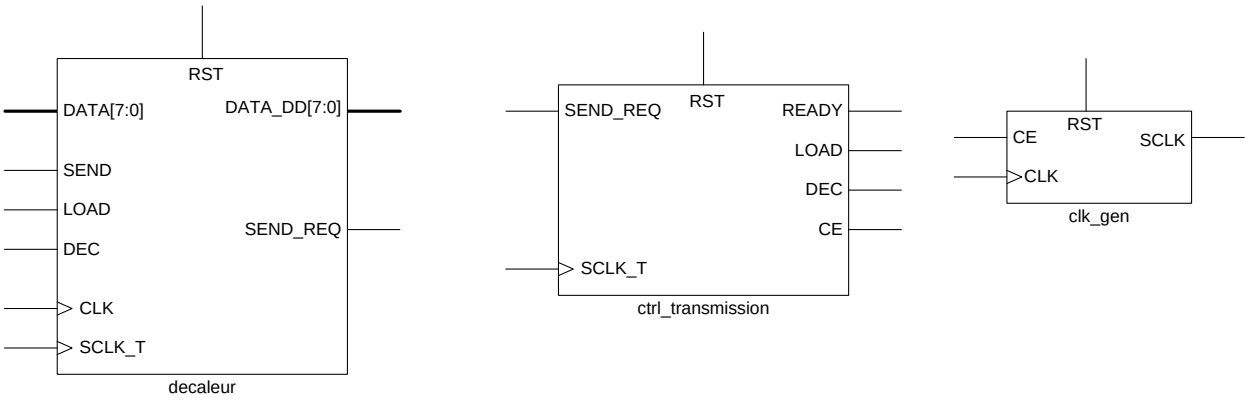


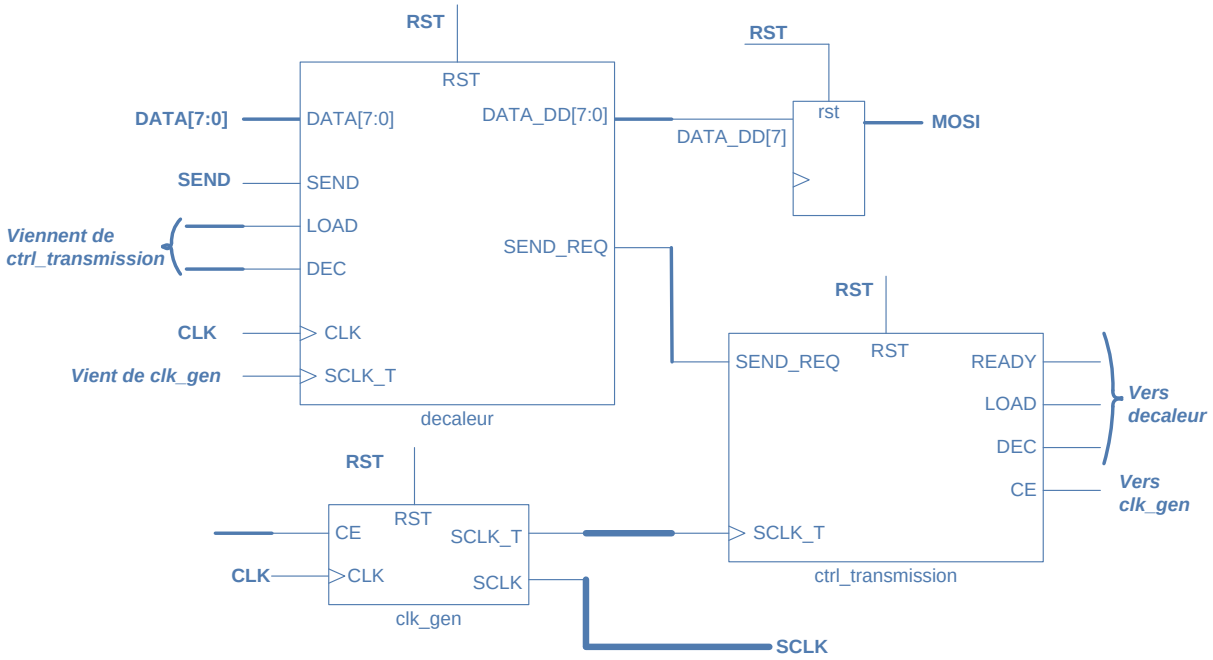
e) Donnez une description VHDL de l'architecture de votre unité de contrôle.

```
architecture arch of ctrl_transmission is

    type state_type is (idle, ld, send);
    signal state : state_type := idle;

begin

    process ( SCLK_T ) is
        variable compteur : natural range 0 to w := 0
    begin
        if ( rising_edge( SCLK_T ) ) then
        if ( RST = '1') then
            state <= idle;
            compteur := 0;
        else
            case state is
                when idle =>
                    compteur := 0;
                    if ( SEND_REQ = '1' ) then
                        state <= ld;
                    end if;
                when ld =>
                    state <= send;
                when send =>
                    compteur := compteur + 1;
                    if( compteur = w ) then
                        state <= idle;
                    end if;
            end case;
        end if;
        end if;
    end process;

    READY <= '1' when ( state = idle and SEND_REQ = '0' ) else '0';
    LOAD <= '1' when state = ld else '0';
    DEC <= '1' when state = send else '0';
    CE <= '1' when state = send else '0';

end architecture;
```

f) Dessinez l'architecture de l'unité de transmission au moyen d'un schéma qui inclue les trois blocs discutés ci-dessus — ceux des questions (b), (c) et (d/e) — ainsi que tout autre composant (portes logiques, mux, registres, etc.) que vous jugerez utile.

BLOCS DE BASE :





Annexe I

```
--
-- processeur.vhd
--
-- processeur à usage général
--
-- Pierre Langlois
-- v. 1.9 2008/11/01 pour labo 5, INF3500 automne 2008
-- version donnée aux étudiants
--
-- Par rapport aux notes de cours (v. 1.53), cette version comporte les changements
suivants:
-- 1. inclut le chargement d'une valeur immédiate de 16 bits.
-- OP code: 1010 | registre-destination | - | -
-- 2. reset asynchrone au lieu de synchrone
-- 3. sortie externe gardée dans un registre
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity processeurv19 is
   generic (
      Nreg : integer := 16; -- nombre de registres
      Wd : integer := 16; -- largeur du chemin des données en bits
      Wi : integer := 16; -- largeur des instructions en bits
      Mi : integer := 8; -- nombre de bits d'adresse de la mémoire d'instr.
      Md : integer := 8; -- nombre de bits d'adresse de la mémoire des données
      resetvalue : std_logic := '1'
   );
   port(
      reset : in std_logic;
      CLK : in std_logic;
      entreeExterne : in signed(Wd - 1 downto 0);
      entreeExterneValide : in std_logic;
      sortieExterne : out signed(Wd - 1 downto 0);
      sortieExterneValide : out std_logic
   );
end processeurv19;

architecture arch of processeurv19 is

-- signaux de la mémoire des instructions
type memoireInstructions_type is array (0 to 2 ** Mi - 1)
   of std_logic_vector(Wi - 1 downto 0);
constant memoireInstructions : memoireInstructions_type :=
   (x"A700", x"0005", x"AC00", x"0003", x"137C", others => (others => '1'));

-- signaux de la mémoire des données
type memoireDonnees_type is array(0 to 2 ** Md - 1) of signed(Wd - 1 downto 0);
signal memoireDonnees : memoireDonnees_type;
signal sortieMemoireDonnees : signed(Wd - 1 downto 0);
signal adresseMemoireDonnees : integer range 0 to 2 ** Md - 1;
signal lectureEcritureN : std_logic;

-- signaux du bloc des registres
type lesRegistres_type is array(0 to Nreg - 1) of signed(Wd - 1 downto 0);
signal lesRegistres : lesRegistres_type;
signal A : signed(Wd - 1 downto 0);
signal choixA : integer range 0 to Nreg - 1;
signal B : signed(Wd - 1 downto 0);
signal choixB : integer range 0 to Nreg - 1;
```

```
signal donnee : signed(Wd - 1 downto 0);
signal choixCharge : integer range 0 to Nreg - 1;
signal charge : std_logic;

-- signaux du multiplexeur contrôlant la source du bloc des registres
signal constante : signed(Wd - 1 downto 0);
signal choixSource : integer range 0 to 3;

-- signaux de l'UAL
signal F : signed(Wd - 1 downto 0);
signal Z : std_logic;
signal N : std_logic;
signal op : integer range 0 to 7;

-- signaux de l'unité de contrôle
type type_etat is
   (depart, querir, decoder, stop, ecrireMemoire,
      lireMemoire, opUAL, jump, chargeimm16);
signal etat : type_etat;
signal PC : integer range 0 to (2 ** Mi - 1); -- compteur de programme
signal IR : std_logic_vector(Wi - 1 downto 0); -- registre d'instruction

begin

   -- multiplexeur pour choisir la source du bloc des registres
   process (F, constante, entreeExterne, sortieMemoireDonnees, choixSource)
   begin
      case choixSource is
         when 0 => donnee <= F;
         when 1 => donnee <= constante;
         when 2 => donnee <= entreeExterne;
         when 3 => donnee <= sortieMemoireDonnees;
         when others => donnee <= F;
      end case;
   end process;
   constante <= signed(memoireInstructions(PC)); -- chargement de 16 bits

   -- bloc des registres
   process (CLK, reset)
   begin
      if reset = resetvalue then
         lesRegistres <= (others => (others => '0'));
      elsif rising_edge(CLK) then
         if charge = '1' then
            lesRegistres(choixCharge) <= donnee;
         end if;
      end if;
   end process;

   -- signaux de sortie du bloc des registres
   A <= lesRegistres(choixA);
   B <= lesRegistres(choixB);

   -- unité de contrôle
   process (CLK, reset)
   begin
      if reset = resetvalue then
         etat <= depart;
      elsif rising_edge(CLK) then
         case etat is
            when depart =>
               PC <= 0;
               etat <= querir;
```

```
            when querir =>
               IR <= memoireInstructions(PC);
               PC <= PC + 1;
               etat <= decoder;
            when decoder =>
               if (IR(15) = '0') then
                  etat <= opUAL;
               else
                  case IR(14 downto 12) is
                     when "000" => etat <= lireMemoire;
                     when "001" => etat <= ecrireMemoire;
                     when "010" => etat <= chargeImm16;
                     when "100" => etat <= jump;
                     when "111" => etat <= stop;
                     when others => etat <= stop;
                  end case;
               end if;
            when opUAL | lireMemoire | ecrireMemoire =>
               etat <= querir;
            when jump =>
               if (IR(11 downto 8) = "0000") or
                  (IR(11 downto 8) = "0001" and Z = '1') or
                  (IR(11 downto 8) = "0010" and Z = '0') or
                  (IR(11 downto 8) = "0011" and N = '1') or
                  (IR(11 downto 8) = "0100" and N = '0')
               then
                  PC <= to_integer(unsigned(IR(7 downto 0)));
               end if;
               etat <= querir;
            when chargeImm16 =>
               etat <= querir;
               PC <= PC + 1;
            when stop =>
               etat <= stop;
            when others =>
               etat <= depart;
         end case;
      end if;
   end process;

   -- signaux de sortie de l'unité de contrôle
   adresseMemoireDonnees <= to_integer(unsigned(IR(7 downto 0)));
   lectureEcritureN <= '0' when etat = ecrireMemoire else '1';
   with etat select
      choixSource <=
         0 when opUAL,
         1 when chargeImm16,
         3 when others;
   choixCharge <= to_integer(unsigned(IR(11 downto 8)));
   choixA <= to_integer(unsigned(IR(7 downto 4)));
   choixB <= to_integer(unsigned(IR(11 downto 8))) when etat = ecrireMemoire else
      to_integer(unsigned(IR(3 downto 0)));
   with etat select
      charge <=
      '1' when opUAL | lireMemoire | chargeImm16,
      '0' when others;
   sortieExterneValide <= '0';
   op <= to_integer(unsigned(IR(14 downto 12)));
```

```
-- UAL
   process(A, B, op)
   begin
      case op is
         when 0 => F <= A + B;
         when 1 => F <= A - B;
         when 2 => F <= shift_right(A, 1);
         when 3 => F <= shift_left(A, 1);
         when 4 => F <= not(A);
         when 5 => F <= A and B;
         when 6 => F <= A or B;
         when 7 => F <= A;
         when others => F <= (others => 'X');
      end case;
   end process;

   -- registre d'état de l'UAL
   process(clk, reset)
   begin
      if reset = resetvalue then
         Z <= '0';
         N <= '0';
      elsif rising_edge(clk) then
         if (etat = opUAL) then
            if F = 0 then Z <= '1'; else Z <= '0'; end if;
            N <= F(F'left);
         end if;
      end if;
   end process;

   -- mémoire des données
   process (CLK)
   begin
      if rising_edge(CLK) then
         if lectureEcritureN = '0' then
            memoireDonnees(adresseMemoireDonnees) <= B;
         end if;
      end if;
   end process;
   sortieMemoireDonnees <= memoireDonnees(adresseMemoireDonnees);

end arch;
```