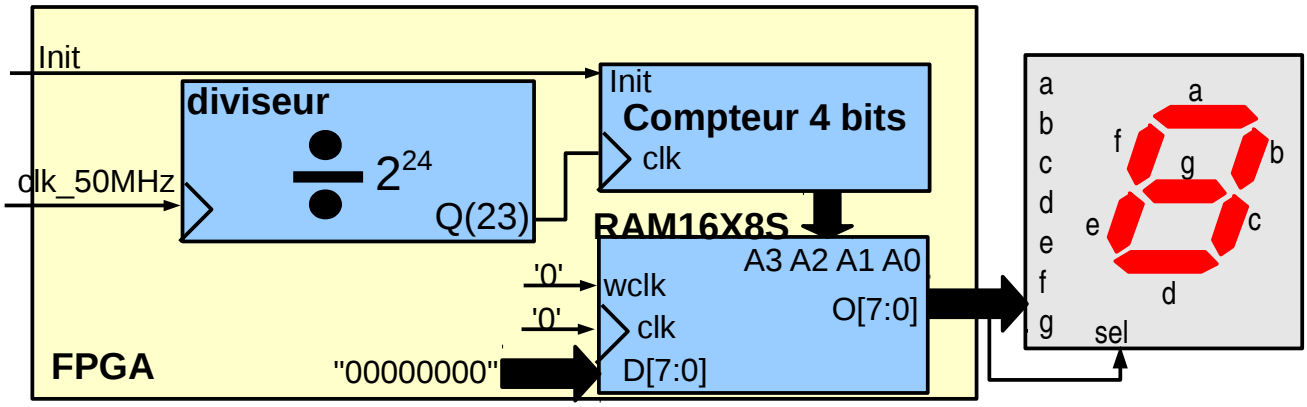
NOM : **Feuille réponse n°1** DS MCENSL1 Nov. 2012

Prénom :
Parcours :

Correction complétée en rouge

Exercice 1

On désire réaliser l'ensemble ci-dessous :



Le cœur de cet ensemble est un compteur binaire sur 4 bits suivi d'une mémoire pour le transcodage (**RAM16X8S**). L'horloge de ce compteur sera réalisée par le poids fort d'un compteur 24 bits : ce compteur sera appelé **diviseur** dans la suite.

I) Étude du diviseur

Le diviseur comporte une entrée appelée "clk_50MHz" et une sortie "clk_slow" appelée "Q(23) sur le schéma. Son entité est donc :

```
1   entity div is
2     port( clk_50Mhz : in std_logic;
3           clk_slow : out std_logic);
4   end div;
```

1°) Compléter l'architecture ci-dessous pour un bon fonctionnement de ce diviseur

Réponse :

```
1   architecture adiv of div is
2   signal cmpt : std_logic_vector(23 downto 0);
3   begin
4     process(clk_50MHz) begin
5         if rising_edge(clk_50MHz) then                    -- compléter ici
6            cmpt <= cmpt + 1 ;                             -- compléter ici
7          end if ;                                         -- compléter ici
8                                                           -- compléter ici
9                                                           -- compléter ici
10                                                          -- compléter ici
11    end process;
12     clk_slow <= cmpt(23) ;                               -- compléter ici
13                                                          -- compléter ici
14  end adiv;
```

2°) Calculer la fréquence de la sortie Q(23)

Réponse : f = 50 000 000 /2^{24} = 2,98 Hz (la division par 2^{24} est représentée dans le dessin)

NOM : **Feuille réponse n°2** DS MCENSL1 Nov. 2012

II) Étude de la mémoire RAM16X8S et de son initialisation

Pour réaliser la mémoire on utilise la mémoire BRAM "**RAM16X8S**" présentée en TD3. Le contenu de cette mémoire est choisi pour afficher 0,1,2,3,4,5,6,7,8,9,A;b,C,d,E,F sur l'afficheur. Quelques indications techniques sont nécessaires pour calculer le contenu de la mémoire :
- pour allumer un segment, il faut un zéro
- g est câblé en poids fort (7), a est câblé en poids (1) et "sel" en poids (0) devra être positionné à 1

Donner les huit valeurs d'initialisation de la mémoire que l'on devra mettre dans la partie "generic map" en complétant la partie réponse ci-après.

Réponse : valeurs hexadécimales en rouges sont identiques aux LUTs correspondantes (non traité en année scolaire 2013/2014)

```
1         generic map (
2         INIT_00 => X"FFFF", --  INIT for bit 0 of RAM
3         INIT_01 => X"2812", --  INIT for bit 1 of RAM
4         INIT_02 => X"D860", --  INIT for bit 2 of RAM
5         INIT_03 => X"D004", --  INIT for bit 3 of RAM
6         INIT_04 => X"8492", --  INIT for bit 4 of RAM
7         INIT_05 => X"02BA", --  INIT for bit 5 of RAM
8         INIT_06 => X"208E", --  INIT for bit 6 of RAM
9         INIT_07 => X"2083") --  INIT for bit 7 of RAM
```

III) Étude du compteur

Le compteur est un composant classique : il compte sur 4 bits possède un signal de RESET appelé "Init" choisi ici asynchrone pour l'initialiser à 0, et bien sûr une horloge.

1°) Écrire l'entité de ce compteur en VHDL en complétant ci-dessous :

Réponse :

```
1         library ieee; -- les librairies sont imposées.
2         use ieee.std_logic_1164.all;
3         use ieee.std_logic_arith.all; -- spécifique à Xilinx
4         use ieee.std_logic_unsigned.all;
5         ENTITY cmpt4 is PORT(
6           clk, Init : in std_logic;
7           q : out std_logic_vector(3 downto 0));
8         END cmpt4
```

2°) Écrire l'architecture de ce compteur en VHDL :

Réponse :

```
1         ARCHITECTURE arch_cmpt4 of cmpt4 is
2         signal cmpt std_logic_vector(3 downto 0);
3         begin
4           process(clk,Init) begin
5             if Init='1' then
6               cmpt <= "0000";
7             elsif rising_edge(clk) then
8               cmpt <= cmpt +1;
9             end if;
10          end process ;
11          q <= cmpt ;
12        end arch_cmpt4 ;
```

Réponse : (suite)

IV Étude de l'ensemble

1°) Donner un nom aux seuls signaux nécessaires à la réalisation du schéma de la figure de la feuille réponse n°1 (à l'aide de "PORT MAP").

Réponse : (sur la figure de la feuille réponse n°1 BIEN LISIBLE SVP)

Un signal est entre Q(23) et clk appelé s_q23 par exemple, un entre cmpt4 et memoire appelé s_q4 par exemple

2°) A partir de l'entité globale ci-dessous et des signaux de la question 1°),

```
1         library ieee; --******************** ENTITE GLOBALE *************
2         use ieee.std_logic_1164.all;
3         entity top is port (
4           clk_50Mhz : in std_logic;
5           aff : out std_logic_vector(3 downto 0);
6           s_7segs : out std_logic_vector(6 downto 0));
7         end top;
```

on vous demande d'écrire les trois PORT MAP de l'architecture globale.

Réponse :

```
1         ARCHITECTURE
2         SIGNAL
3         SIGNAL
4         -- On laisse tomber les composants ! Pas de place !
5         BEGIN
6           i1 : div PORT MAP( clk_50Mhz => clk_50Mhz,
7                              clk_slow => s_q23) ;
8
9           i2 : cmpt4 PORT MAP(clk =>  s_q23,
10                              Init => Init,
11                              q => s_q4) ;
12          i2 :   RAM16X8S PORT MAP(wclk =>'0', clk => '0', D => "00000000",
13                 A3 => s_q4(3), A2 => s_q4(2), A1 => s_q4(1), A0 => s_q4(0),
14                 O => s_7segs) ;
15          aff <= "1110" ;
16        END aTop;
```

V) Réalisation de l'ensemble avec un picoBlaze (A partir de 2013 utilisation ATMega8 embarqué on ne corrige donc pas cette partie !!!!)

Le diviseur est maintenant réalisé par le poids fort de l'unique PORT de sortie d'un picoBlaze. On donne plus loin (page suivante) le programme qui réalise cela. Lisez-le en vous aidant des commentaires.

1°) Que vaut 0xB8 en décimal ?

Réponse :

Toutes les instructions du picoBlaze sont exécutées en 2 fronts d'horloge. L'horloge a une fréquence de 50 MHz.

Tous les résultats des questions suivantes seront donnés en secondes ou millisecondes et en nombre de fronts d'horloge.

2°) Combien de temps dure l'exécution de la boucle interne entre les lignes 18 et 20 ?

Réponse :

NOM : **Feuille réponse n°4** DS MCENSL1 Nov. 2012

```
1          NAMEREG s7,i
2          NAMEREG s6,j
3          NAMEREG s4,cmpt
4          NAMEREG s0,max
5
6                  LOAD  max,B8     ; met 0xB8 dans max qui ne changera plus
7                  LOAD cmpt,00     ; met 0x00 dans cmpt
8          loop:   OUTPUT cmpt,00   ; sort cmpt dans l'unique port
9                  CALL  wait       ; appelle sous-programme d'attente
10                 ADD   cmpt,01    ; ajoute 1 à cmpt
11                 jump  loop       ; retour à loop
12         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
13         ;=== double boucle d'attente===
14         wait:
15                 LOAD i,max       ;chargement de max=0xB8 dans i
16         w:
17                 LOAD j,max       ;chargement de max=0xB8 dans j
18         w_1:
19                 SUB j,01         ; retire 1 à j
20                 JUMP NZ, w_1     ; tant que pas zéro on boucle à w_1
21           SUB i,01               ; retire 1 à i
22           JUMP NZ,w              ; tant que pas zéro on boucle à w
23           RETURN                 ; fin du sous-programme "wait"
```

3°) En déduire combien de temps dure l'exécution de la boucle externe (+ la boucle interne) entre les lignes 14 et 22 ?
Réponse :

4°) En déduire la fréquence du bit de poids fort du seul et unique PORT du picoBlaze ?
Réponse : (avec quelques explications)

Exercice 2 Étude d'un miniprocesseur embarqué "MCPU"

Tim Boscke, a publié en 2001-2004 un miniprocesseur de quatre instructions (http://opencores.org/project,mcpu) destiné à tenir dans un CPLD.

Cette architecture utilise une seule mémoire pour les données et le programme. Données et programme sont donc mélangés dans une seule et même mémoire.

1°) Le compteur programme destiné à chercher les instructions en mémoire programme est sur 6 bits. Quelle est la taille de la mémoire programme sachant que, comme le montre le tableau suivant, les opcodes (instructions en binaire) sont sur 8 bits ?

Réponse : 2^6 octets = 64 octets

Les instructions sont les suivantes :

Mnémonique	***Opcode***	***Description***
NOR	00AAAAAA	Accu = Accu NOR mem[AAAAAA]
ADD	01AAAAAA	Accu = Accu + mem[AAAAAA] + maj retenue
STA	10AAAAAA	mem[AAAAAA] = Accumulateur
JCC	11DDDDDD	positionne PC à DDDDDD quand retenue =0 + clear carry

2°) Dans le tableau ci-dessus, AAAAAA désigne une adresse mémoire en binaire sur 6 bits tandis que DDDDDD désigne une destination sur aussi 6 bits.

Quelle est la taille de la mémoire donnée adressable par AAAAAA ?

Réponse : 2^6 octets = 64 octets (idem à tout à l'heure car mémoire commune)

3°) L'astuce de cette architecture est qu'il est possible de définir par dessus ces instructions des **macros** instructions (ensemble d'une ou plusieurs instructions) de manière assez subtile :

Macro	***Assembleur***	***Description***
CLR	NOR allone	Accu = 0 (allone doit contenir0xFF)
LDA mem	NOR allone,ADD mem	Charge la memoire dans l'accumulateur
NOT	NOR zero	inverse le contenu de l'accumulateur (zero contient 0x00
JMP dst	JCC dst, JCC dst	Saut inconditionnel à dest
JCS dst	JCC *+2, JCC dst	Saut si retenue
SUB mem	NOR zero,ADD mem, ADD one	Soustrait mem de l'accu (one contient 0x01)

Vous disposez à ce point de 9 instructions.
Pouvez-vous donner l'opcode (code binaire) de "JMP dst" ?
Réponse : JCC dst, JCC dst soit 11DDDDDD 11DDDDDD

4°) Le code source de ce processeur est tellement court que nous le donnons maintenant pour analyse succincte.

```
1         library ieee;
2         use ieee.std_logic_1164.all;
3         use ieee.std_logic_unsigned.all;
4
5         entity CPU8BIT2 is
6                 port (  data:   inout   std_logic_vector(7 downto 0);
7                         adress: out     std_logic_vector(5 downto 0);
8                         oe:     out     std_logic;
9                         we:     out     std_logic;      -- Asynchronous memory
          interface
10                        rst:    in      std_logic;
11                        clk:    in      std_logic);
12        end;
13
14        architecture CPU_ARCH of CPU8BIT2 is
15          signal  akku:   std_logic_vector(8 downto 0);   -- akku(8) is carry !
16          signal  adreg:  std_logic_vector(5 downto 0);
17          signal  pc:     std_logic_vector(5 downto 0);
18          signal  states: std_logic_vector(2 downto 0);
19        begin
20          process(clk,rst)
21            begin
22              if (rst = '0') then
23            -- start execution at memory location 0
```

```
24                adreg   <= (others => '0');
25                states  <= "000";
26                akku <= (others => '0');
27                pc   <= (others => '0');
28              elsif rising_edge(clk) then
29            -- PC / Adress path
30                if (states = "000") then
31                  pc      <= adreg + 1;
32                  adreg   <= data(5 downto 0);
33                else
34                  adreg   <= pc;
35                end if;
36             -- ALU / Data Path
37                case states is
38                -- add
39                  when "010" => akku <= ("0" & akku(7 downto 0)) + ("0" & data);
40                -- nor
41                  when "011" => akku(7 downto 0) <= akku(7 downto 0) nor data;
42                  when "101" => akku(8) <= '0';-- branch not taken, clear carry
43                -- instr. fetch, jcc taken (000), sta (001)
44                  when others => null;
45                end case;
46            -- State machine
47          -- l'operateur VHDL "/=" signifie différent de
48                if (states /= "000") then states <= "000";  -- fetch next opcode
49                elsif (data(7 downto 6) = "11" and akku(8)='1') then
50                  states <= "101";   -- branch not taken
51                else  states <= "0" & not data(7 downto 6);-- execute instruction
52                end if;
53              end if;
54            end process;
55            -- output
56            adress  <= adreg;
57            data    <= "ZZZZZZZZ" when states /= "001" else akku(7 downto 0);
58            oe <= '1' when (clk='1' or states  = "001" or rst='0' or
59                 states = "101") else '0'; -- no memory access during reset and
60                                           -- state "101" (branch not taken)
61            we <= '1' when (clk='1' or states /= "001" or rst='0') else '0';
62          end CPU_ARCH;
```

4-a) Le signal "Akku" représente le registre Accumulateur. **CPU8bit2** étant une architecture 8 bits on s'attend à ce que celui-ci soit sur 8 bits. Pourquoi est-il sur 9 bits au lieu de 8 ?

Réponse : ligne 42 when "101" => akku(8) <= '0';-- branch not taken, clear carry

nous laisse penser que le 9° bit est la cary (retenue en français)

4-b) La machine d'états qui séquence le processeur est décrite entre la ligne 47 et la ligne 51. Son état est décrit par le signal "state". Combien d'états (au maximum) peut comporter cette machine ?

Réponse : ligne 18 : signal states: std_logic_vector(2 downto 0);

nous laisse penser qu'on est sur 3 bits donc que l'on a 8 états possibles.

4-c) Quel est l'état futur de l'état "001" ?

Réponse : ligne 48 : if (states /= "000") then states <= "000"; dit que l'état futur est « 000 »

Exercice 3

On désire réaliser un circuit capable de remplir une mémoire à partir d'un fichier HEX envoyé sur une liaison série. Nous allons étudier partiellement ce circuit. HEX est un format texte, on

veut donc le transformer en hexadécimal. Par exemple, "FE" sera transformé en 0xFE où "FE" représente une donnée sur 16 bits (2 codes ASCII consécutifs 0x46 pour 'F' et 0x45 pour 'E'). Le circuit qui fait cette transformation est un circuit combinatoire appelé **comb**.

I) Étude du transcodage (comb)

Le circuit de transcodage est un circuit combinatoire dont on donne le code VHDL ci-dessous :

```
1         library ieee;
2         use ieee.std_logic_1164.all;
3         entity comb is port (
4           data : in std_logic_vector(7 downto 0);
5           value_bit,colon_bit,LF_bit,CR_bit : out std_logic;
6           value4 : out std_logic_vector(3 downto 0)
7           );
8         end comb;
9
10        architecture acomb of comb is
11        signal valueMoins30,valueMoins37 :  std_logic_vector(7 downto 0);
12        signal sortie : std_logic_vector(7 downto 0);
13        begin
14          valueMoins30 <= data - x"30";
15          valueMoins37 <= data - x"37";
16          sortie <= "1000" & valueMoins30(3 downto 0) when
17                                    (data <= x"39" and data >=x"30") else
18                    "1000" & valueMoins37(3 downto 0) when
19                                    (data <= x"46" and data >=x"41") else
20                    "0100" & x"F" when data = x"3A" else --colon = ":"
21                    "0010" & x"F" when data = x"0D" else --LF Line Feed
22                    "0001" & x"F" when data = x"0A" else --CR Carriage Return
23                    "0000" & x"F"; -- Error
24          value4 <= sortie(3 downto 0);
25          value_bit <= sortie(7);
26          colon_bit <= sortie(6);
27          LF_bit <= sortie(5);
28          CR_bit <= sortie(4);
29        end acomb;
```

1°) Calculer les valeurs des signaux valueMoins30 et valueMoins37 si on a le caractère 'B' en entrée (de code ASCII 0x42)

Réponses : 0x42 – 0x30 = 0x12, et 0x42 – 0x37 = 0x0B

2°) Calculer la valeur du signal "sortie" de ce composant si on a le caractère 'B' en entrée

Réponse : sortie est "1000" & X"B" (& est concaténation) soit "10001011" en binaire

3°) Calculer toutes les sorties de ce composant si on a le caractère 'B' en entrée

Réponses : value4 est X"B", value_bit est '1', et le reste à '0'

II) Le composant combinatoire précédent est connecté à un composant FIFO qui reçoit les données provenant de la RS232. On donne l'entité complète de ce FIFO :

```
1         entity bbfifo_16x9 is
```

NOM : **Feuille réponse n°8** DS MCENSL1 Nov. 2012

```
2            Port (       data_in : in std_logic_vector(8 downto 0);
3                        data_out : out std_logic_vector(8 downto 0);
4                           reset : in std_logic;
5                           write : in std_logic;
6                            read : in std_logic;
7                            full : out std_logic;
8                        half_full : out std_logic;
9                    data_present : out std_logic;
10                            clk : in std_logic);
11       end bbfifo_16x9;
```

1°) Parmi ces entrées et sorties laquelle connecteriez vous au composant combinatoire **comb** de I) ? Quel problème cela pose-t-il ?

Réponses : "data_in" relié à value4. Le problème est qu'elles ne sont pas de la même taille.

2°) Le code source du FIFO nous montre l'extrait suivant :

```
1             count_lut: LUT4
2             generic map (INIT => X"6606")
3             port map( I0 => pointer(0),
4                       I1 => read,
5                       I2 => pointer_zero,
6                       I3 => write,
7                        O => half_count(0));
```

On vous demande de faire la table de vérité correspondante et d'écrire son équation correspondante.

Réponses : Karnaugh donne I3./I1.I0 + I3.I1./I0 + /I2./I1.I0 + /I2.I1./I0

I3	I2	I1	I0	0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0