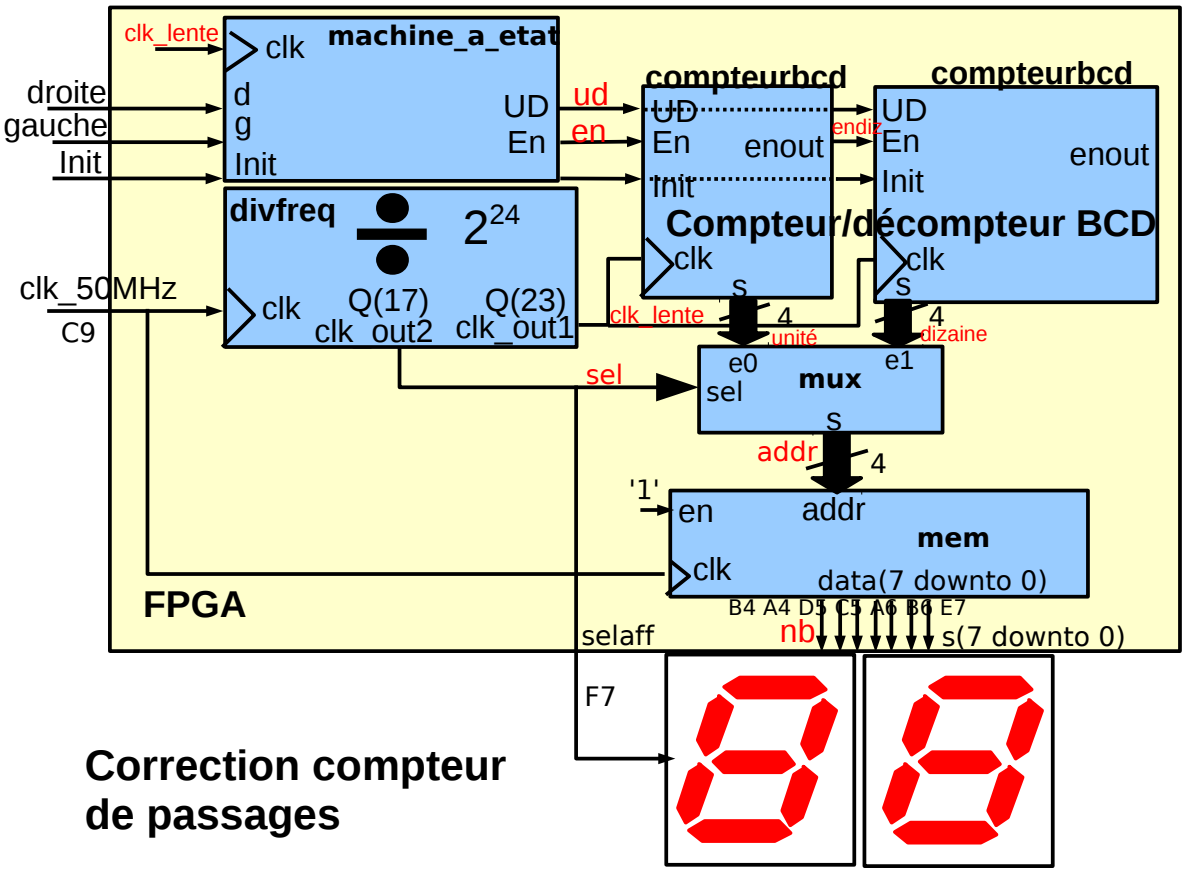
NOM : **Feuille réponse n°1** DS MCENSL1 Nov. 2013

Prénom :

Groupe : **CORRECTION AJOUTEE EN ROUGE**

Exercice 1

On désire étudier partiellement le compteur de passages ci-dessous :



Correction compteur de passages

Cet ensemble a été réalisé en première année mais différemment.

I) Transcodeur

Le composant nommé "**mem**" dans le schéma ci-dessus est un transcodeur binaire vers 7 segments. On désire le remplacer par un composant purement combinatoire donc sans horloge et sans entrée "en".

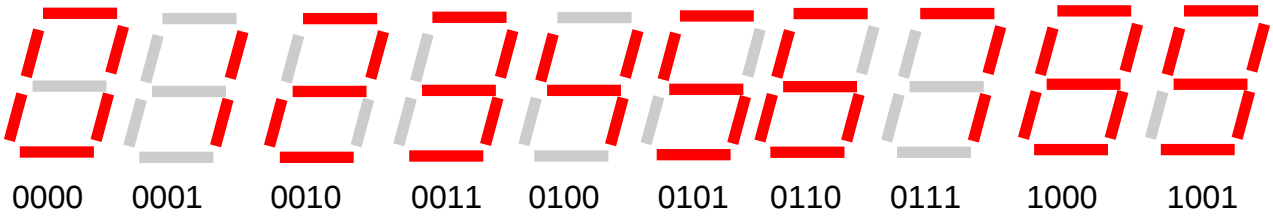
1°) Écrire la nouvelle entité.

Réponse :

```
ENTITY mem IS PORT(
  addr : IN std_logic_vector(3 downto 0) ;
  data : out std_logic_vector(6 downto 0) ;
END mem ;
```

NOM : **Feuille réponse n°2** DS MCENSL1 Nov. 2013

2°) Écrire l'architecture correspondante pour afficher correctement



et un tiret ("g" allumé) pour les autres cas.

Réponse :

```
ARCHITECTURE arch OF mem IS
BEGIN
  WITH addr SELECT
    data <= "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",
            "0000001" WHEN OTHERS ;
END arch ;
```

II) Étude du diviseur (divfreq)

Le diviseur comporte une entrée appelée "clk" et, deux sorties "clk_out1" et "clk_out2" reliée respectivement à "Q(23)" et "Q(17)" sur le schéma. Son entité est donc :

```
1   entity divfreq is
2     port( clk : in std_logic;
3         clk_out1, clk_out2 : out std_logic);
4   end divfreq;
```

NOM : **Feuille réponse n°3** DS MCENSL1 Nov. 2013

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

Réponse :

```
1   architecture adiv of divfreq is
2   signal cmpt : std_logic_vector(23 downto 0);
3   begin
4     process(clk) begin
5         if clk'event and clk='1' 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_out1 <= cmpt(23) ;                          -- compléter ici
13     clk_out2 <= cmpt(17) ;                          -- compléter ici
14                                                     -- compléter ici
15  end adiv;
```

2°) Calculer la fréquence de la sortie Q(17) qui sélectionne l'afficheur. Respecte-t-on les 25 Hz minimum qui évitent tout clignotement visible à l'œil ?

Réponses : f=50 000 000/ 2^18 = 190,73 Hz ==> pas de clignotement visible à l'oeil

III) Étude du compteur/décompteur BCD

Le compteur décompteur BCD est un composant cascadable : il compte/décompte 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 cmptBCD IS PORT (
6     clk,init,en,ud : in std_logic ;
7     s:out std_logic_vector(3 downto 0) ;
8     enout : out std_logic) ;
9   END cmpt_BCD ;
10
11
12
```

2°) On donne l'architecture correspondante :

```
1   architecture behavior of cmptbcd is
2     signal n : std_logic_vector(3 downto 0);
3     begin
4     increment : process(clk) begin
5           if clk'event and clk='1' then
6                 if init ='1' then
7                       n <= (others => '0');
```

```
8                     elsif en='1' then
9                           if ud = '1' then  --up
10                                if n<9 then
11                                      n <= n + 1 ;
12                                else
13                                      n <= (others => '0');
14                                end if;
15                          else -- down
16                                if n>0 then
17                                      n <= n - 1 ;
18                                else
19                                      n <= "1001";
20                                end if;
21                          end if;
22                    end if;
23              end if;
24        end process;
25        enableout: process(n,en,ud)
26        begin
27              if en='1' then
28                    if ud='1' and n=9 then
29                          enout<= '1';
30                    elsif ud='0' and n=0 then
31                          enout<='1';
32                    else
33                          enout<='0';
34                    end if;
35              else -- pour eviter comptage intempestif sur dizaine
36                enout<='0';
37              end if;
38        end process;
39        s <= n;
40      end behavior;
```

Quelle valeur doit-on mettre en "ud" pour réaliser un comptage ?

Réponse : ud='1' (lignes 9, 10, 11 et 12)

Quel est l'état futur du compteur s'il est positionné à 13 et que "ud" est à 0 (et "en" à 1). Est-ce un fonctionnement normal pour un compteur décompteur BCD ?
Réponses : 12 fonctionnement anormal pour du BCD mais en principe on n'arrive jamais en 13 !

Exercice 2 (Étude partielle d'un cœur CORDIC)

Dans cet exercice, on cherche à interfacer un cœur CORDIC (capable de calculer des sinus et cosinus) à notre ATMega8.

I) Étude du format Q3.13

Les architectures 8 bits (comme l'ATMega8) ne sont pas ou peu adaptées aux calculs en virgule flottante (type "float" du langage c). Ces types sont en effet définis sur 32 bits. Pour remédier à ce problème, nous allons utiliser un format en virgule fixe sur 16 bits, que nous allons étudier maintenant.

NOM : **Feuille réponse n°5** DS MCENSL1 Nov. 2013

Nous décidons d'utiliser un format Q3.13 qui veut dire que trois bits de poids forts sont la partie entière et les treize bits restants sont la partie fractionnaire. Voici un résumé du codage utilisé avec les poids correspondants :

b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
S	2	1	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}

S est le signe et la partie entière possède donc 2 bits, l'ensemble est codé en complément à deux.

1°) Calculer les valeurs de 1, 0.5 et 0.25 (sur 16 bits en binaire).

Réponses :

	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
1 =	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0.5=	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0.25=	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

2°) Calculer -0.25 en format Q3.13 en complément à deux (on complémente 0.25 bit à bit et on lui ajoute 1)

Réponse : (sur 16 bits en binaire) /0.25 est le complément bit à bit auquel on ajoute +1

	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
/0.25=	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
+1=	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

3°) Lorsqu'on passe de la valeur 0.5 à la valeur 0.25 on multiplie par $2^{-1} = ½$. Quelle opération sur un registre (décalage gauche, décalage droit, décalage gauche circulaire, décalage droit circulaire) est alors réalisée ? Quelle valeur supplémentaire est alors entrée dans le registre (0 ou 1) ?

Réponses : décalage droit avec entrée de '0' sur la gauche

4°) Lorsqu'on passe de la valeur -0.5 à la valeur -0.25 on multiplie par $2^{-1} = ½$. Quelle opération sur un registre (décalage gauche, décalage droit, décalage gauche circulaire, décalage droit circulaire) est alors réalisée ? Quelle valeur supplémentaire est alors entrée dans le registre (0 ou 1) ?

Réponses :

	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
-0.5=	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

décalage droit avec entrée de '1' sur la gauche

II) Étude d'un compteur d'angle

Pour faciliter l'utilisation d'un composant CORDIC, on décide de réaliser un compteur/décompteur en format Q3.13 que l'on interfacera plus tard à notre processeur

NOM : **Feuille réponse n°6** DS MCENSL1 Nov. 2013

ATMega8. Voici l'entité de ce compteur.

```
1         ENTITY cmpt_Angle is
2           PORT(
3             clk,reset,en : IN std_logic;
4             inc_dec_i : IN std_logic_vector(7 DOWNTO 0);
5             -- 3243 < angle_o < CDBD
6             angle_o : OUT std_logic_vector(15 downto 0));
7         END cmpt_Angle;
```

1°) Les valeurs données en commentaire sont les valeurs extrémales (en hexadécimal) du domaine d'angles $\{\frac{-\pi}{2}, \frac{\pi}{2}\}$ valides pour le cœur CORDIC en format Q3.13. Donnez les valeurs correspondantes en binaire.

Réponses :

$\frac{-\pi}{2}$ = 0XCDBD (car négatif) $\frac{\pi}{2}$ = 0x3243 (positif)

2°) Ces deux valeurs sont-elles complément à deux l'une de l'autre ?

Réponse : 0x3243 → complément à 1 ->0xCDBC → +1 → 0xCDBD Ils sont bien complément à deux l'un de l'autre ce qui est tout à fait normal !

3°) Pour ne pas se tracasser avec des histoires de signe, on décide de raisonner directement sur les valeurs hexadécimale et de définir dans l'architecture deux constantes :

```
1         ARCHITECTURE Behavioural OF cmpt_Angle IS
2         constant MAX : std_logic_vector(15 downto 0) := x"????";
3         constant MIN : std_logic_vector(15 downto 0) := x"????";
4         -- =1 si incrementation =0 si decrementation
```

Donnez les valeurs à mettre en lieu et place des '?'

Réponses : MAX = x"CDBD " et MIN = x"3243" car on ne tient pas compte du signe

4°) Voici partiellement le code du compteur d'angle

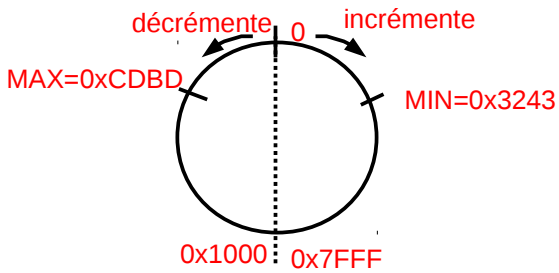
```
1         PROCESS(clk,reset) BEGIN
2             IF reset='1' THEN
3               s_angle <=MAX;
4               increment <= '1';
5               ELSIF rising_edge(clk) THEN
6                 IF en = '1' THEN
7                   IF increment = '1' THEN
8                     s_Angle <= s_Angle + inc_dec_i;
9                     IF s_Angle >= MIN and s_Angle(15)='0' THEN
10                       s_angle <= MIN;
11                       increment <= '0';
12                    END IF;
13                  ELSE
14                    s_Angle <= s_Angle - inc_dec_i;
15                   IF s_Angle <= MAX and s_Angle(15)='1' THEN
16                       s_angle <= MAX;
17                       increment <= '1';
18              END IF;
```

Pouvez-vous expliquer pourquoi dans le code du compteur on trouve dans la partie incrémentation (lignes 7 à 13) une remise à MIN (ligne 10) si on a un dépassement au lieu

d'une remise à MAX ?

Réponse : (difficile donc en bonus)

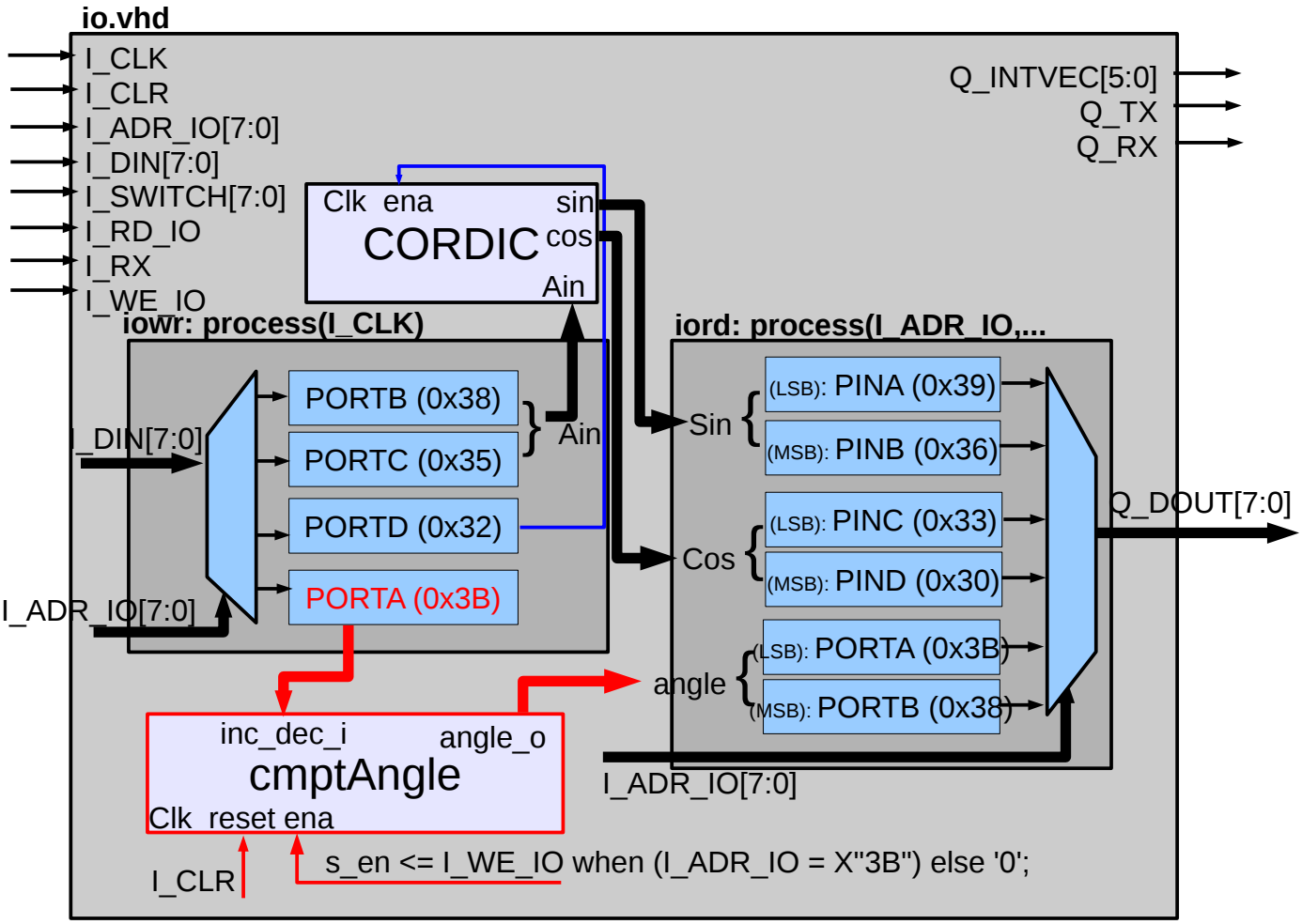
Rien de tel qu'une représentation circulaire



III) Interfacer le compteur d'angle à l'ATMega8

Nous en arrivons maintenant à l'interfaçage de ce compteur d'angle à l'ATMega8.

Le schéma de principe est donné ci-dessous. Seule le partie inférieure du dessin où l'on trouve le compteur d'angle "cmptAngle" nous intéresse. Dans ce schéma LSB désigne le poids faible et MSB le poids fort.



1°) Quelle instruction C permet d'incrémenter ou décrémenter un angle si cette opération est réalisée en mettant une valeur dans "inc_dec_i" ?

Réponse : PORTA = 0xFF ; //ajoute ou retire 0xFF au compteur d'angle

2°) Quelle(s) instruction(s) C permet(tent) de lire la valeur de l'angle retournée si celle-ci est disponible dans "angle_o" en précisant la déclaration de variable(s) correspondante(s) ?

Réponses : uint_8_t angleLSB, angleMSB ;

angleLDB = PORTA ; angleMSB = PORTB ;

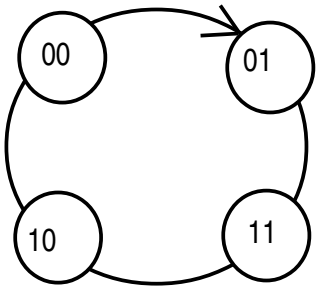
3°) Quelle(s) instruction(s) C supplémentaire(s) permet(tent) alors de mettre le résultat dans une seule variable de type "uint16_t" ?

Réponses :

```
uint_16_t angle ;
angle = angleMSB ;
angle <<= 8 ;
angle = angle + angleLSB ;
```

Exercice 3 (VHDL classique)

Donner un programme VHDL qui implante le diagramme d'évolution ci-dessous sans utiliser de "case when", c'est à dire avec des équations de récurrences :



Réponse :

```
1   library ieee; -- les librairies ieee
2   use ieee.std_logic_1164.all;
3   ENTITY gray IS PORT(
4     clk : in std_logic ;
5     q : out std_logic_vector(1 downto 0)) ;
6   END gray ;
7
8   ARCHITECTURE agray of gray IS
9   SIGNAL cmptGRAY : std_logic_vector(1 downto 0) ;
10  BEGIN
11    PROCESS(clk) BEGIN
12      IF clk'event and clk='1' then
13        cmptGRAY(1) <= cmptGRAY(0) ;
14        cmptGRAY(0) <= NOT cmptGRAY(1) ;
15      END IF ;
16    END PROCESS ;
17    q <= cmptGRAY ;
18  END agray ;
```