Systèmes embarqués
TD N°04 : VHDL

Exercice 1

Soit le code VHDL suivant d'un compteur simple :

```
Library ieee;
Use ieee.std_logic_1164.all;
 Use ieee.numeric_std.all;
 Use ieee.std_logic_unsigned.all;
entity CMP4BITS is
PORT ( CLOCK : in std_logic;
Q : inout std_logic_vector(3 downto 0));
end CMP4BITS;
architecture DESCRIPTION of CMP4BITS is
 begin process (CLOCK)
begin
 if (CLOCK ='1' and CLOCK'event) then
Q <= Q + 1;
 end if;
 end process;
end DESCRIPTION;
```

1) Expliquer ce code. Pourquoi nous avons utilisé : ieee.numeric_std.all et ieee.std_logic_unsigned.all

   Il s'agit d'un compteur simple sur 4 bits. Le déclenchement du ***process*** se fait sur un changement d'état du signal ***CLOCK***, l'incrémentation de la sortie ***Q*** se fait sur le front montant de l'horloge ***CLOCK***.

   L'incrémentation du compteur est réalisée par l'opérateur **+** associé à la valeur entière **1**. Or le type de Q est un vecteur std_logic. On ne peut pas par défaut additionner un nombre entier **1** avec un bus de type électronique (***std_logic_vector***), c'est pour cela que l'on rajoute dans la partie déclaration des bibliothèques les lignes:

   Use ieee.numeric_std.all;

   Use ieee.std_logic_unsigned.all;

   Ces deux bibliothèques ont des fonctions de conversions de types et elles permettent d'associer un entier avec des signaux électroniques.

2) Effectuer les modifications sur le code si Q était « out » au lieu de « inout »

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
entity CMP4BITS is
PORT (
```

```
CLOCK : in std_logic;
Q : out std_logic_vector (3 downto 0));
end CMP4BITS;
architecture DESCRIPTION of CMP4BITS is
signal CMP : std_logic_vector(3 downto 0);
begin
process (CLOCK)
-- ce qui est en commentaire si vous voulez que l'incrémentation se
--fait chaque deux périodes
-- Variable inc: integer:=0;
begin
if (CLOCK ='1' and CLOCK'event) then
-- if(inc mod 2=0) then
CMP <= CMP + 1;
-- end if;
-- inc:=inc+1;
end if;

end process;
-- affectation du bus interne au signal de sortie Q
Q <= CMP;
end DESCRIPTION;
```

3) Donner le code VHDL d'un compteur 3 bits avec remise à zéro asynchrone

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
entity CMP3BITS is
PORT (
CLOCK : in std_logic;
RESET : in std_logic;
Q : out std_logic_vector(2 downto 0));
end CMP3BITS;
architecture DESCRIPTION of CMP3BITS is
signal CMP: std_logic_vector (2 downto 0);
begin
process (RESET,CLOCK)
begin
if RESET ='1' then
CMP <= "000";
elsif (CLOCK ='1' and CLOCK'event) then
CMP <= CMP + 1;
end if;
end process;
Q <= CMP;
end DESCRIPTION;
```

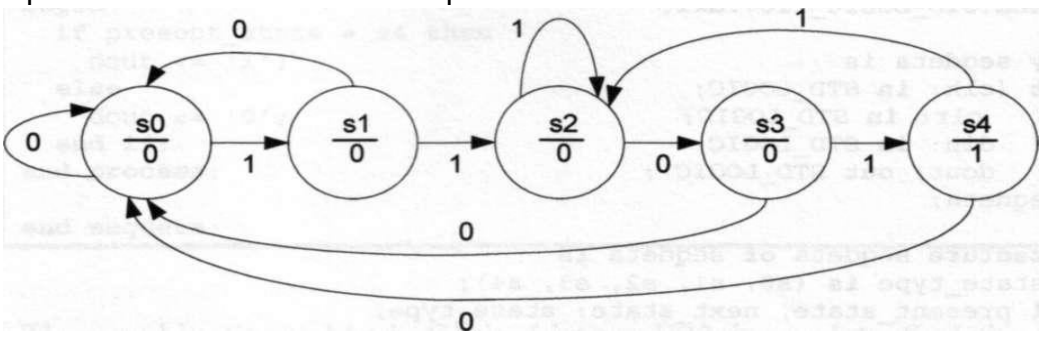
4) Donner le code VHDL d'un compteur 3 bits avec remise à zéro synchrone

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;
entity CMP3BITS is
```

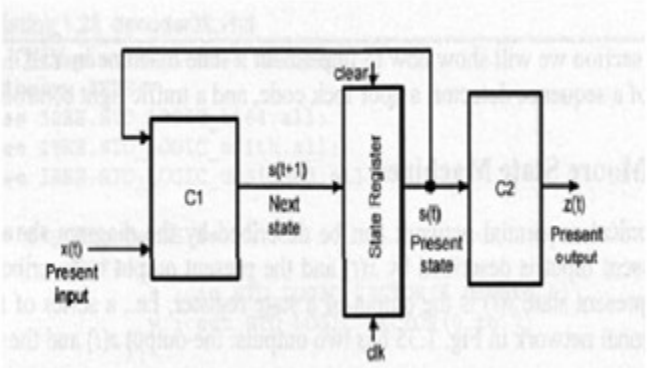
```
PORT (
CLOCK : in std_logic;
RESET : in std_logic;
Q : out std_logic_vector (2 downto 0));
end CMP3BITS;
architecture DESCRIPTION of CMP3BITS is
signal CMP: std_logic_vector (2 downto 0);
begin
process (CLOCK)
begin
if (CLOCK ='1' and CLOCK'event) then
if RESET ='1' then
CMP <= "000";
else
CMP <= CMP + 1;
end if;
end if;
end process;
Q <= CMP;
end DESCRIPTION;
```

Exercice 2 :

Soit le graphe d'état d'un détecteur de séquence suivante :



1) Moore ou Mealy ? Quelle est la séquence binaire détectée ?
   Moore. Séquence détectée : 1101
2) Coder en VHDL la machine de Moore?



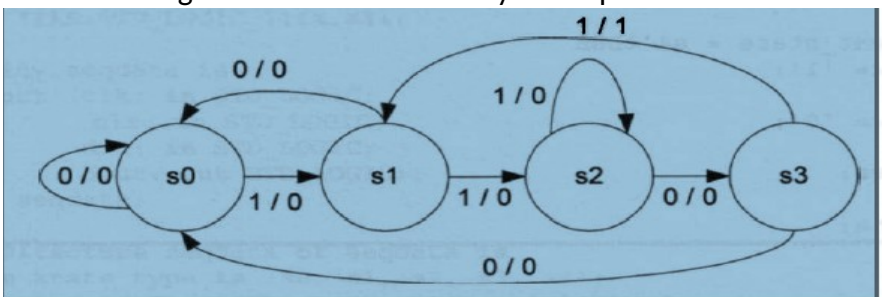
-- Solution classique avec trois process

```
Library ieee;
Use ieee.std_logic_1164.all;
```

```
entity seqdeta is
PORT (
CLk, clear, din : in std_logic;
dout : out std_logic);
end seqdeta;
architecture seq of seqdeta is
type State is (s0,s1,s2,s3,s4);
Signal present_state, next_state: State;
begin
 --- processus de mémorisation états
 mem_state: process(clk,clear)
   begin
   if clear='1' then
     present_state <=s0;
   elsif clk'event and clk='1' then
     present_state<=next_state;
   end if;
 end process mem_state;

 -- processus combinatoire état futur
 maj_state: process (din, present_state)
 begin
 case present_state is
   when s0 =>
     if din='1' then
     next_state <=s1;
     else
     next_state <=s0;
     end if;
   when s1 =>
     if din='1' then
     next_state <=s2;
     else
     next_state <=s0;
     end if;
   when s2 =>
     if din='0' then
     next_state <=s3;
     else
     next_state <=s2;
     end if;
   when s3 =>
     if din='1' then
     next_state <=s4;
```

```
    else
    next_state <=s0;
    end if;
   when s4 =>
    if din='0' then
    next_state <=s0;
    else
    next_state <=s2;
    end if;
   when others =>
    null;
  end case;
  end process maj_state;

  -- processus combinatoire sortie
  maj_sortie: process (present_state)
  begin
   if (present_state = s4) then
    dout<='1';
    else
     dout<='0';
     end if;
  end process maj_sortie;

 end seq ;
```

3) Donner le diagramme d'états de Mealy correspondant?



4) Coder en VHDL la machine de Mealy ?

Solution classique de trois process	Solution en deux process
Library ieee; Use ieee.std_logic_1164.all; entity seqdetb is PORT (CLk, clear, din : in std_logic; dout : out std_logic); end seqdetb; architecture seq of seqdetb is	Library ieee; Use ieee.std_logic_1164.all; entity seqdetb is PORT (CLk, clear, din : in std_logic; dout : out std_logic); end seqdetb; architecture seq of seqdetb is

```
type State is (s0,s1,s2,s3);
Signal present_state, next_state: State;
begin
 --- processus de mémorisation états
mem_state: process(clk,clear)
  begin
  if clear='1' then
    present_state <=s0;
  elsif clk'event and clk='1' then
    present_state<=next_state;
  end if;
end process mem_state;

-- processus combinatoire état futur
maj_state: process (din, present_state)
begin
case present_state is
  when s0 =>
    if din='1' then
    next_state <=s1;
    else
    next_state <=s0;
    end if;
  when s1 =>
    if din='1' then
    next_state <=s2;
    else
    next_state <=s0;
    end if;
  when s2 =>
    if din='0' then
    next_state <=s3;
    else
    next_state <=s2;
    end if;
  when s3 =>
    if din='1' then
    next_state <=s1;
    else
    next_state <=s0;
    end if;
  when others =>
    null;
end case;
```

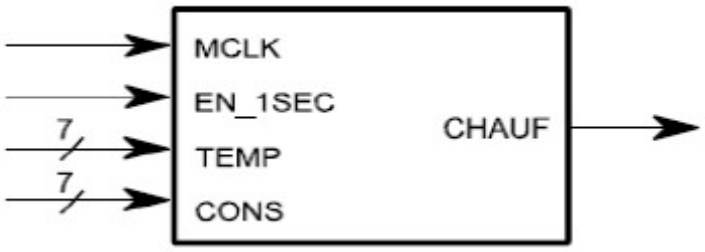
```
type State is (s0,s1,s2,s3);
Signal present_state, next_state: State;
begin
 --- processus de mémorisation états
mem_state: process(clk,clear)
  begin
  if clear='1' then
    dout <='0';
    present_state <=s0;
  elsif clk'event and clk='1' then
    present_state<=next_state;
  end if;
end process mem_state;

-- processus combinatoire
comb: process (din, present_state)
begin
case present_state is
  when s0 =>
    dout <='0';
    if din='1' then
    next_state <=s1;
    else
    next_state <=s0;
    end if;
  when s1 =>
    dout <='0';
    if din='1' then
    next_state <=s2;
    else
    next_state <=s0;
    end if;
  when s2 =>
    dout <='0';
    if din='0' then
    next_state <=s3;
    else
    next_state <=s2;
    end if;
  when s3 =>
    if din='1' then
    dout <='1';
    next_state <=s1;
    else
```

```
 end process maj_state;

 -- processus combinatoire sortie
 maj_sortie: process (din, present_state,clear)
 begin
  if clear ='1' then
    dout <= '0';
  elsif (present_state = s3) and din ='1' then
    dout<='1';
    else
      dout<='0';
      end if;
 end process maj_sortie;

end seq ;
```

```
    dout <='0';
    next_state <=s0;
    end if;
   when others =>
    null;
 end case;
 end process comb;


end seq ;
```

Exercice 3 : Circuit de régulation de température

Un capteur de température fournit la température ambiante sur 7 bits. Le but est de maintenir la température d'une pièce à une consigne donnée. Une sortie sera appliquée (tout ou rien) à un circuit de chauffage pour le faire fonctionner ou l'arrêter. On appellera **CHAUF** cette sortie (figure 1).



On veut concevoir une machine à 3 états finis qui opère comme suit :

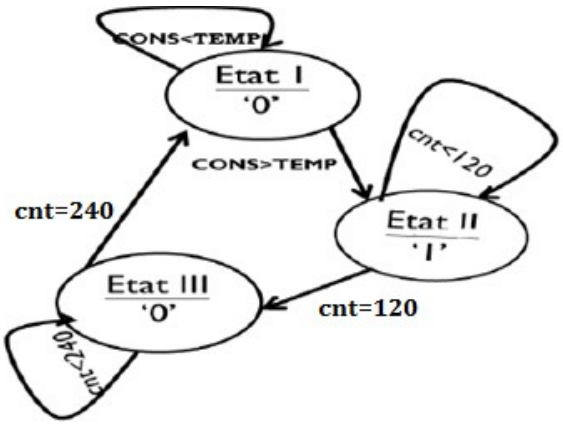
Etat1- On mesure la température (appelée **TEMP**) et on la compare avec la consigne (appelée **CONS**).

Etat2- Dans le cas où une différence positive est constatée (consigne plus chaude que la pièce), on envoie '1' en sortie "**CHAUF**" pendant une durée de 2 minutes.

Etat3- A la fin de cette durée, on se met dans l'état attente pendant 4 minutes où la sortie reste à '0' et ensuite on repart à l'état1 où une nouvelle lecture de température sera faite.

L'interface externe du circuit de contrôle est la présentée en figure. Les ports **MCLK**, **EN_1SEC** et **CHAUF** sont codés sur 1 bit. On suppose que l'entrée EN_1SEC soit délivrée par un temporisateur externe qui génère une impulsion de 1 cycle à chaque seconde, il sert pour le calcul des durées des étapes 2 et 3.

1) Schématiser le graphe d'états de ce système



2) Donner le code VHDL correspondant au système.

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
entity temp_control is
PORT (
CLk, en1sec : in std_logic;
temp, cons: in std_logic_vector(6 downto 0);
chauf : out std_logic);
end temp_control;

architecture fsm of temp_control is
type Tstate is (etat1,etat2,etat3);
Signal present_state, next_state: Tstate;
begin
 --- processus de mémorisation états
mem_state: process(clk)
  begin
  if clk'event and clk='1' then
    present_state<=next_state;
  end if;
 end process mem_state;

 -- processus state_sortie
 proc: process (en1sec, present_state)
 variable diff: std_logic_vector(6 downto 0);
 variable cmpt: Integer:=0;
 begin
  case present_state is
  when etat1 =>
    chauf <='0';
    diff:=cons-temp;
    if diff(6)='0' then
    next_state <=etat2;
```

```
      else
      next_state <=etat1;
      end if;
    when etat2 =>
      chauf <='1';
      if (en1sec ='1') then
      cmpt :=cmpt+1;
      end if;
      if cmpt=120 then
      cmpt :=0;
      next_state <= etat3;
      else
      next_state <=etat2;
      end if;
    when etat3 =>
      chauf <='0';
      if (en1sec ='1') then
      cmpt :=cmpt+1;
      end if;
      if cmpt=240 then
      cmpt :=0;
      next_state <= etat1;
      else
      next_state <=etat3;
      end if;
     when others =>
      null;
  end case;
  end process proc;

end fsm ;
```