Examen VHDL 2eme session 2009

Exercice 1 : Détecteur de parité (3 points)

On souhaite modéliser un détecteur de parité pour des mots de 4 bits par une description VHDL qui est composée de 2 parties :

- Interface (entité) :

§ Signal d'entrée (mode in) : DIN (données de type bit_vector 4 bits)

§ Signaux de sortie (mode out): ODP (parité impaire de type bit), EVP (parité paire de type bit)

- Comportement (architecture) :

§ ODP= '1' si le nombre de '1' dans DIN est impaire, sinon ODP= '0'

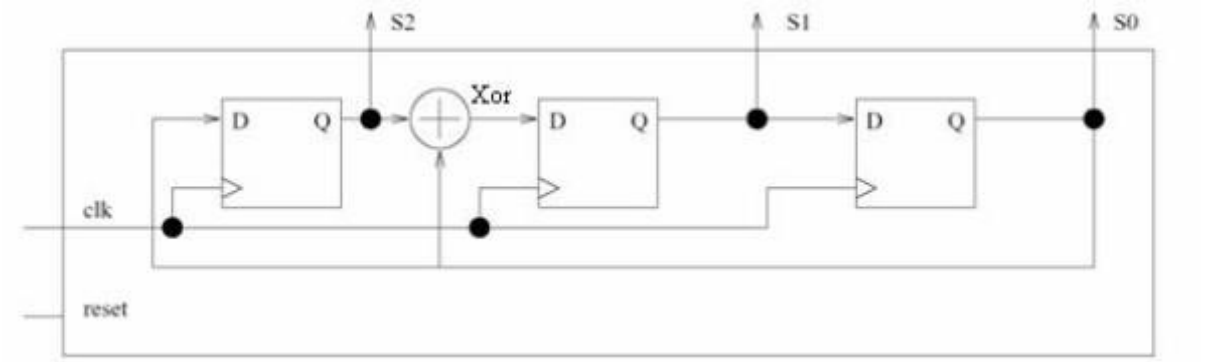
§ EVP='1' si le nombre de '1' dans DIN est pair, sinon EVP= '0'

a- Donner la table de vérité selon le 15 différentes valeurs possibles de DIN.

b- Coder le comportement avec l'instruction d'assignation sélective (with…. Select).

Exercice 2 : (3 points)

Registres à décalage à rétroaction linéaire, on souhaite utiliser le registre à décalage à rétroaction linéaire (normalement employé pour générer des bits aléatoires) suivant pour réaliser un compteur:



Ecrivez une description comportementale en VHDL (entité et architecture) qui réalise ce circuit.

Exercice 3 : (3 points)

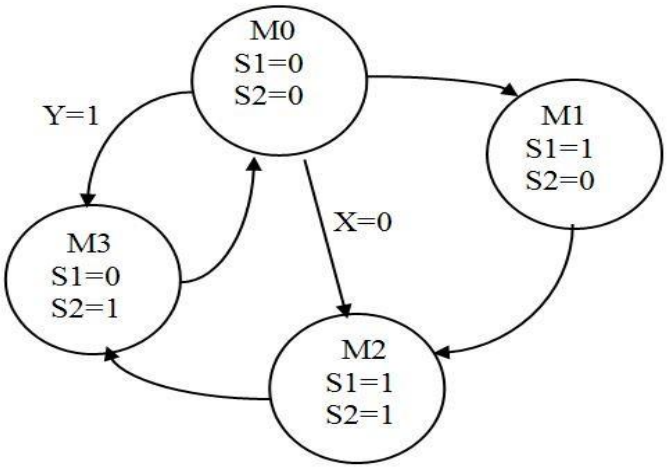
On considère la fonction (incremente) ci-contre:

```
function incremente (e : in bit_vector)
return bit_vector is
    variable s : bit_vector(e'left downto 0) ;
    variable c : bit ;
begin
    c := e(0) ;
    s(0) := not c ;
    for i in 1 to e'left loop
        s(i) := c xor e(i) ;
        c := c and e(i) ;
    end loop ;
    return s ;
end incremente ;
```

1. Dans quelles parties d'un programme VHDL peut on insérer un tel code ? (Deux réponses au moins)
2. Représenter le schéma généré par cette fonction si on l'applique à un vecteur de trois éléments binaires.
3. Quelle est l'opération réalisée ?
4. Utiliser cette fonction pour réaliser un compteur binaire synchrone 16 bits qui compte à chaque front montant d'horloge.
5. Modifier le compteur précédent pour lui adjoindre une commande **raz de remise à zéro synchrone** et une commande **d'autorisation de comptage EN**.

Exercice 4 : (4 points)

La description d'état est utilisée pour décrire des systèmes séquentiels quelconques (machine d'état). La description du système se fait par un nombre fini d'états. Ci-dessous la représentation schématique d'un système à 4 états (M0 à M3), 2 sorties (S1 et S2), 2 entrées X et Y, sans oublier l'entrée d'horloge qui fait avancer le processus, et celle de remise à zéro qui permet de l'initialiser :



L'état initial est M0. Les 2 sorties sont à 0. Au coup d'horloge on passe inconditionnellement à l'état M1 sauf si la condition Y=1 a été vérifiée, ce qui mène à l'état M3 ou si X=0 a été validé ce qui mène à M2.
De M3 on revient au coup d'horloge à M0. De M1 on passe à M2, et de M2 on passe à M 3...
Dans chaque état on définit les niveaux des sorties.

Ecrire la description VHDL (Entité et Architecture) de ce système en se basant sur le diagramme d'état. Le système est actif sur front montant.

Solution

Exercice 1 :

```
Library ieee;
use ieee.std_logic_1164.all;
entity parity is port
        (D_IN: in std_logic_vector (3 downto 0);
        ODP,EVP : out std_logic);
end parity;
architecture ARCH_parity of parity is
signal S:std_logic_vector (1 downto 0);
Begin
with D_IN select
S<="00" when "0000",
  "10" when "0001",
  "10" when "0010",
  "01" when "0011",
  "10" when "0100",
  "01" when "0101",
  "01" when "0110",
  "10" when "0111",
  "10" when "1000",
  "01" when "1001",
  "01" when "1010",
  "10" when "1011",
```

```
   "01" when "1100",
   "10" when "1101",
   "10" when "1110",
   "01" when "1111",
   "00" when others;
ODP<=S(1);
EVP<=S(0);
end ARCH_parity;
```

Exercice 2 :

```
entity compteur
port( clk, reset: in std logic;
S0, S1, S2: out std logic));
end entity compteur;
architecture comport of compteur is
begin
P1: process(clk, reset)
begin
if reset='1' then
S0 <= '1';
S1 <= '0';
S2 <= '0';
elsif clk='1' and clk'event then
S2 <= S0;
S1 <= S2 xor S0;
S0 <= S1;
end if;
end process P1;
end architecture comport;
```

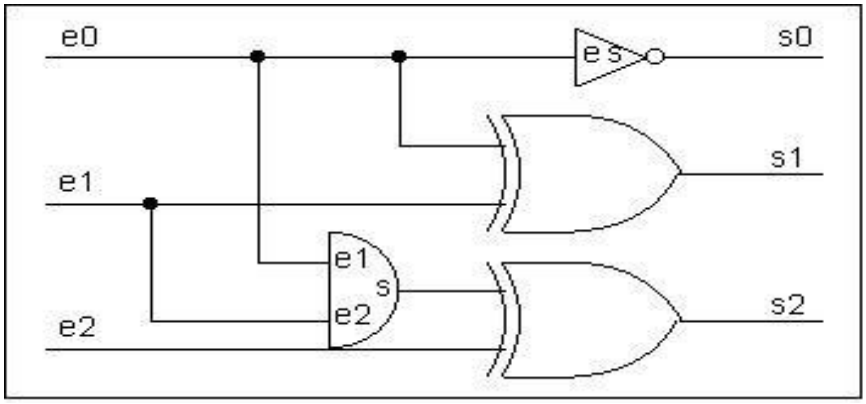
Exercice 3 :

1. La définition d'un sous programme VHDL appartient à une zone déclarative ou à une bibliothèque d'objets ressources. La fonction peut être définie dans :

- La zone déclarative d'une entité,
- La zone déclarative d'une architecture,
- Dans un paquetage (*package*).

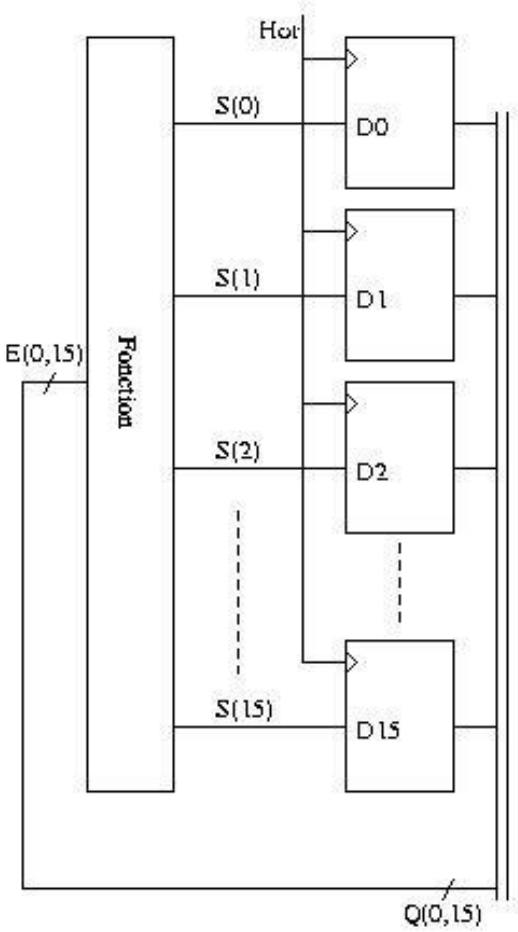
Dans la dernière méthode, qui est de loin la meilleure, la déclaration de la fonction (sa signature, ou l'équivalent d'un prototype du langage C) appartient à l'en-tête de paquetage (*package header*) tandis que la définition (le code) de la fonction appartient au corps (*pacqkage body*) du paquetage.

2.

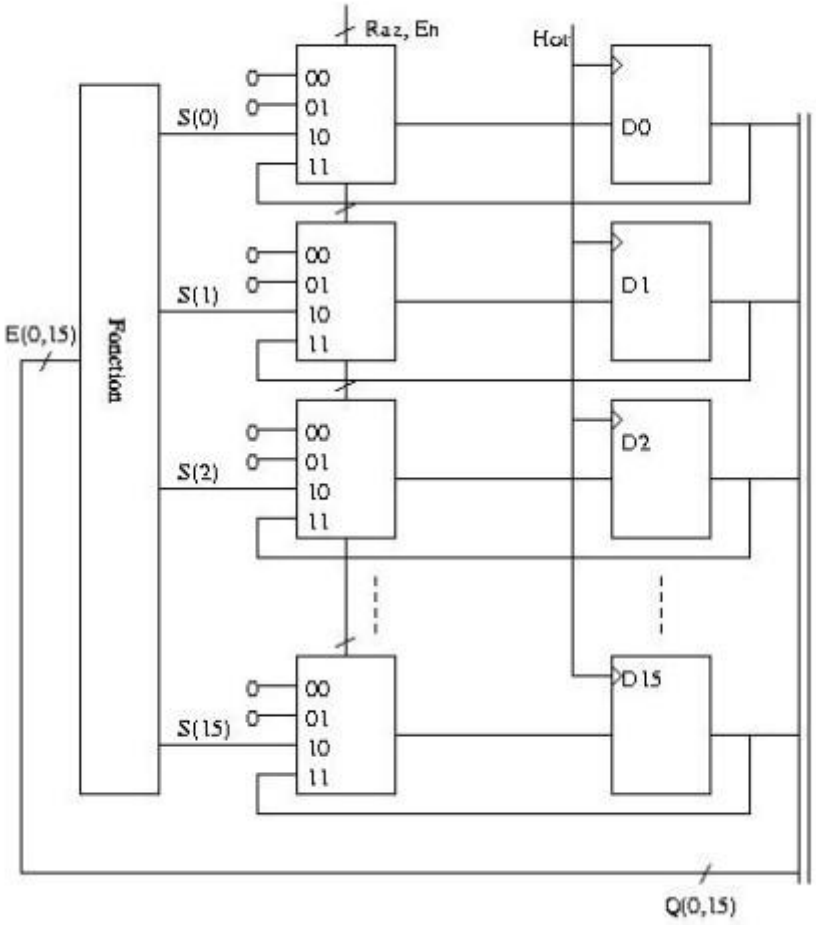


3. La fonction réalisée est un incrémenteur : le vecteur de sortie représente un nombre codé en binaire qui est égal au nombre représenté par le vecteur d'entrée plus 1.

4.
```
entity compteur is
  port(hor : in bit ;
       compte : out bit_vector(15 downto 0)) ;
function incremente (e : in bit_vector)
-- suite de la définition de la fonction
end incremente ;
end compteur ;
architecture fonc of compteur is
  signal cpt : bit_vector(15 downto 0) ;
begin
  compte <= cpt ;
process
begin
  wait until hor = '1' ;
    cpt <= incremente(cpt) ;
end process ;
end fonc ;
```



5.
```
entity compteur is
  port(hor, en, raz : in bit ;
       compte : out bit_vector(15 downto 0)) ;
function incremente (e : in bit_vector)
-- suite de la définition de la fonction
end incremente ;
end compteur ;
architecture fonc of compteur is
  signal cpt : bit_vector(15 downto 0) ;
begin
  compte <= cpt ;
process
begin
  wait until hor = '1' ; -- commandes synchrones
    if raz ='0' then -- mise à zéro prioritaire
      cpt <= (others => '0') ;
    elsif en = '0' then -- comptage actif à '0'
      cpt <= incremente(cpt) ;
    end if ; -- par défaut : mémoire
end process ;
end fonc ;
```




```
6.                    entity                    compteur                    is
  port(hor,         en,         raz         :         in         bit         ;
       compte       :       out       bit_vector(15       downto       0))       ;
function         incremente         (e         :         in         bit_vector)
--       suite       de       la       définition       de       la       fonction
end                                  incremente                                  ;
end compteur ;
architecture             fonc             of             compteur             is
   signal       cpt       :       bit_vector(15       downto       0)       ;
begin
   compte                       <=                       cpt                       ;
process(hor,    raz)    --    deux    signaux    activent    le    processus
begin
      if    raz    ='0'    then    --    mise    à    zéro    asynchrone
         cpt          <=          (others          =>          '0')          ;
      elsif    hor'event    and    hor    =    '1'    then    --    synchrone
         if    en    =    '0'    then    --    comptage    actif    à    '0'
            cpt              <=              incremente(cpt)              ;
         end      if      ;      --      par      défaut      :      mémoire
      end                                  if                                  ;
```

```
end                                    process                                    ;
end fonc ;
```

Exercice 4 :

```
library ieee;
use ieee.std_logic_1164.all;
  ENTITY machine1 IS
PORT (RAZ, horl :IN STD_LOGIC;  -- Ne pas oublier remise à 0 et horloge !
                    X, Y :IN STD_LOGIC;
                   S1, S2 :OUT STD_LOGIC);
END machine1;

ARCHITECTURE diagramme OF machine1 IS
        TYPE etat_4 IS (M0, M1, M2, M3
  SIGNAL etat,etat_suiv :etat_4 := M0;
BEGIN
        definir_etat:PROCESS( raz, horl)
                BEGIN
                        If raz = '1' THEN
                                etat <= M0;
                        ELSIF rising_edge(horl) THEN  etat <= etat_suiv ;
                        END IF;
        END PROCESS definir_etat ;
sorties : process (etat, x, y)
        BEGIN
                CASE etat IS
                                WHEN M0 => S1 <= '0'; S2<= '0';
                                        IF Y='1' then etat_suiv <= M3;
                                          elsif X='0' then etat_suiv <= M2;
                                          ELSE etat_suiv <= M1;
                                        END IF;

                                WHEN M1 => S1 <= '1'; S2<= '0';etat_suiv <= M2;
                                WHEN M2 => S1 <= '1'; S2<= '1';etat_suiv <= M3;
                                WHEN M3 => S1 <= '0'; S2<= '1';etat_suiv <= M0;
                        END CASE;
END process sorties ;
END diagramme ;
```