Examen VHDL 1ere session 2009

Exercice 1 : Comparateur (3 points)

On veut modéliser un circuit permettant de comparer 2 bus de données de 8 bits, A et B, et de générer 3 signaux de sortie :

EQUA si les données sur les deux bus sont égales,

SUPE si la donnée du bus A est supérieure à la donnée du bus B,

INFE si la donnée du bus A est inférieure à la donnée du bus B,

A et B sont des entrées du type std_logic_vecteur;

EQUA, SUPE, INFE sont des sorties de type std_logic ;

Ecrire l'entité de ce comparateur de 8 bits et coder l'architecture de ce modèle.

Exercice 2 : Modélisation et styles en VHDL (3 points)

Soit la description VHDL suivante:

```
entity exercice2 port (
 x1, x2, x3, sel: in std_logic;
y: out std_logic);
end  exercice2;
architecture archi of exercice2 is
signal a, b, c, d, e, f: std_logic;
begin
a <= x1 or x3;
b <= x1 and x3;
c <= x2 and a;
```

```
d <= b or c;

e <= x1 xor x2;

f <= x3 xor e;

P1: process (d, f, sel)

begin

if sel='0' then

y <= d;

else

y <= f;

end if;

end process P1;

end architecture archi;
```

1. Tracez à partir d'éléments de base le schéma correspondant.

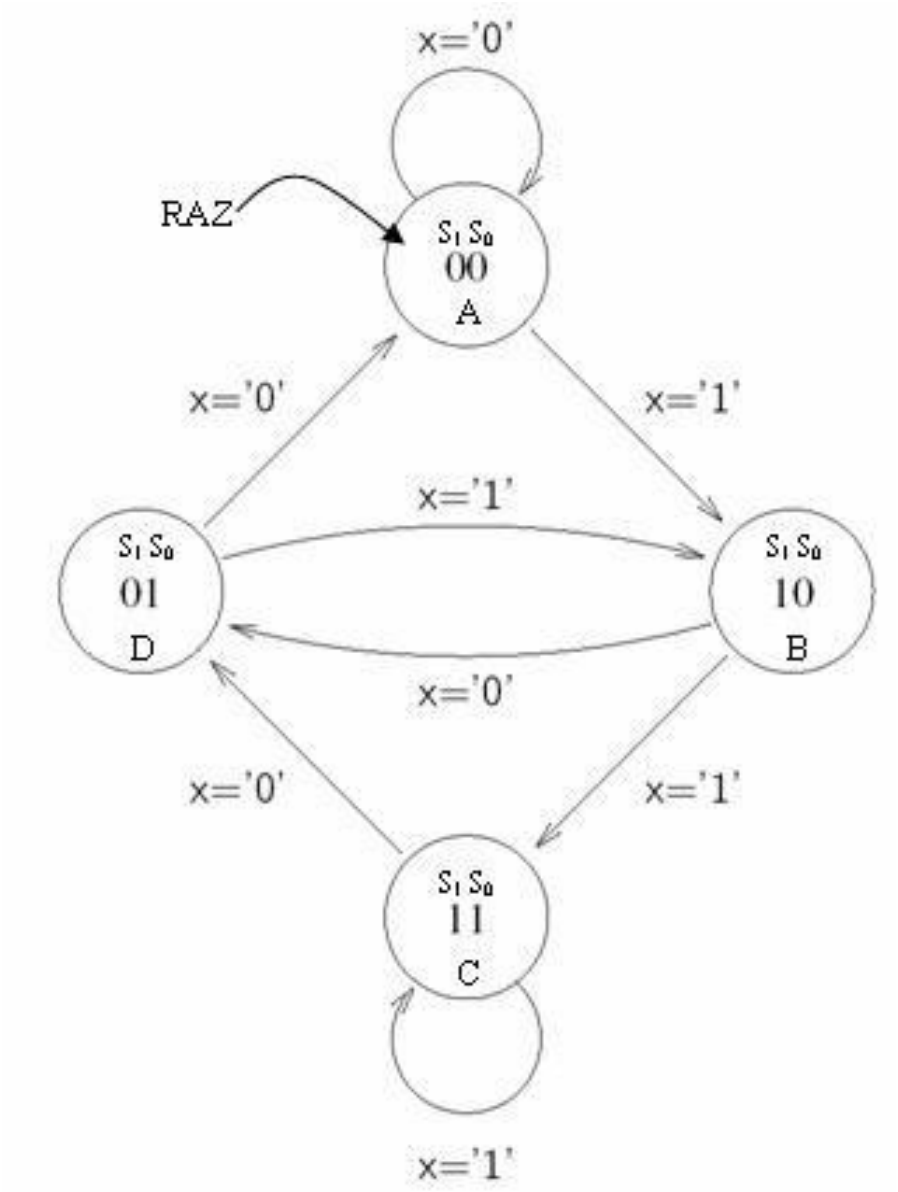
2. Le processus P1 est-il combinatoire ou séquentiel? Justifiez.

3. Quel est selon vous la fonction de ce circuit?

4. À partir de la même entité, écrivez une architecture exploitant la logique 3-etats pour la sélection de la sortie via le signal sel.

Exercice 3 : Diagramme d'état (4 points)

Le système à concevoir dispose de trois entrées et de deux sorties. Les entrées sont l'horloge clk, RAZ et la commande x ; les sorties sont S_0 et S_1. Les entrées et les sorties sont de type std_logic sauf x est de type bit. La description du système se fait par un nombre fini d'états. Ci-dessous la représentation schématique d'un système à 4 états (A, B, C et D) :

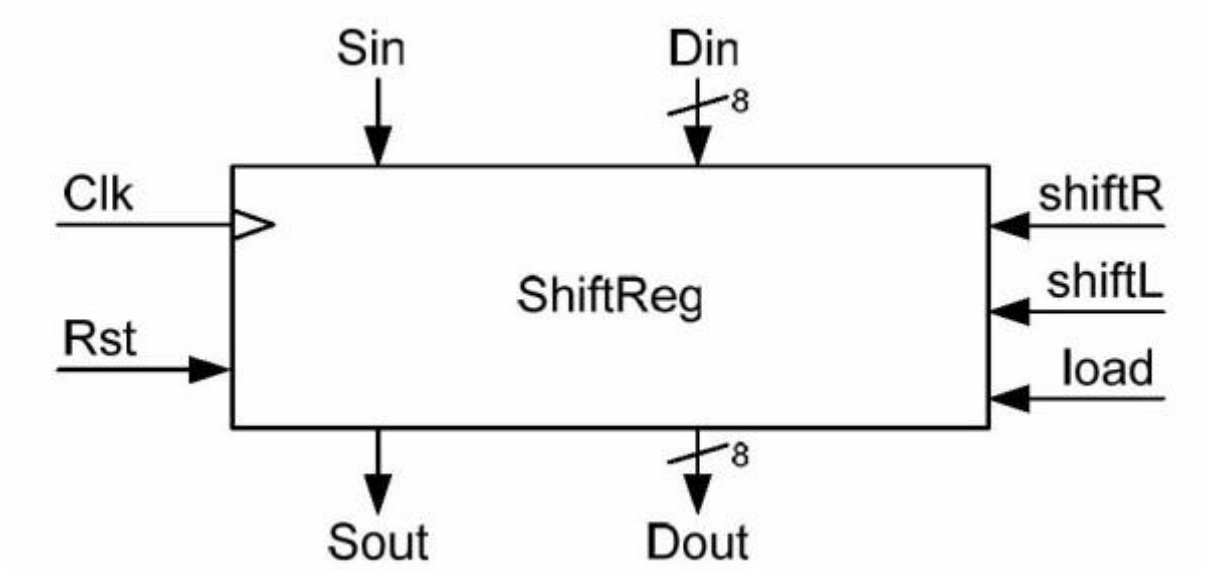


1. Donner la description en langage VHDL (Entité et Architecture) de ce système en se basant sur le diagramme d'état. Le système est actif sur front montant.

2. Compléter le chronogramme suivant du système :

Exercice 4 : Registre à décalage (3 points)

Le registre à décalage commandé est représenté sur la figure suivant. Ce registre doit être capable de réaliser un décalage vers la droite (commandé par le signal **ShiftR**), et un décalage vers la gauche (commandé par le signal **ShiftL)** et utilise des données de 8 bits.



Ce composant accepte deux entrées :

• Une entrée série (**Sin**), dont la valeur est copiée dans le bit de poids fort du registre si le signal **shiftR** vaut **'1',** ou dans le bit de poids faible si **shiftL** vaut **'1'.**

• Une entrée parallèle (**Din**) sur 8 bits, qui est chargée dans le registre sur un front montant d'horloge si le signal **Load** vaut **'1'**.

Le composant dispose également de deux sorties :

• Une sortie série **Sout** qui correspond à la valeur courante du bit de poids le plus faible.

• Une sortie parallèle (**Dout**) sur 8 bits, qui représente la valeur courante du registre.

1. Donner la table de vérité de ce registre à décalage commandé.

1. Donnez la description en VHDL synthétisable de ce registre utilisant un process. Dans le process, vous pouvez utiliser l'instruction if…. Then et l'instruction for…..loop.

Correction

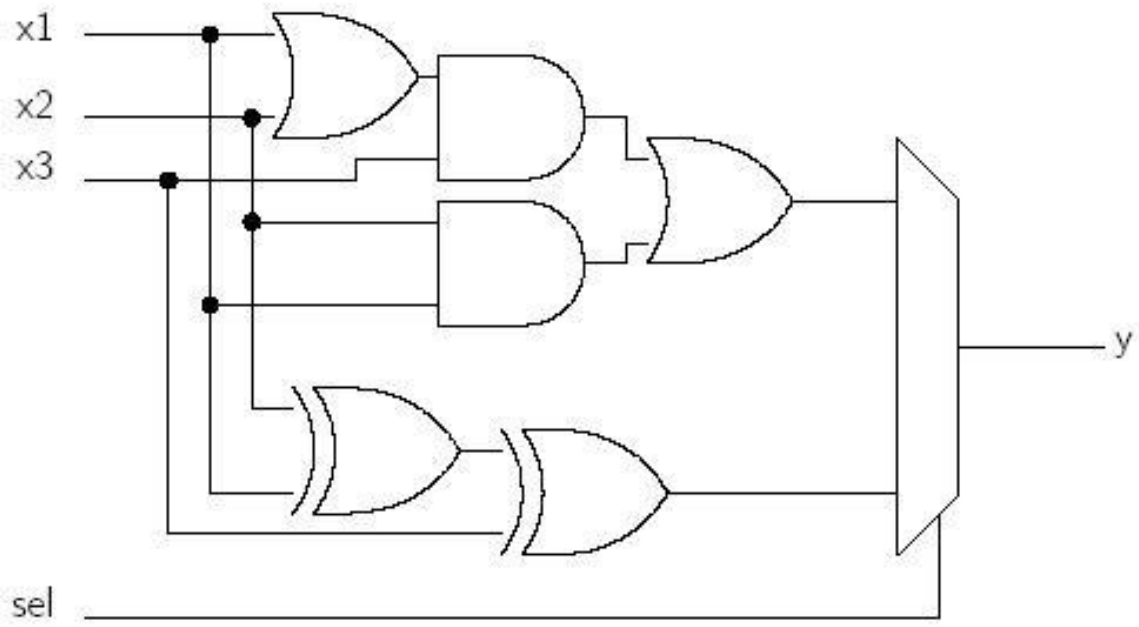
Exercice 1 :

```
Library ieee;
use ieee.std_logic_1164.all;
entity comparateur8bits is port
        (A, B: in std_logic_vector (7 downto 0);
        EQUA, SUPE, INFE: out std_logic);
end comparateur8bits;

architecture ARCH_comparateur8bits of comparateur8bits is
Begin
EQUA<='1' when A=B else '0';
SUPE<='1' when A>B else '0';
INFE<='1' when A<B else '0';
end ARCH_comparateur8bits;
```

Exercice 2 :

1- Le schéma correspondant



2- C’est une description comportementale, puisqu’il n’y a aucune composante.

C’est un processus combinatoire car il inclut toutes les entrées (signaux lus) dans la liste de sensitivité. De plus, toutes les valeurs possibles de sel sont traitées.

3 - C’est un plein additionneur avec multiplexage des sorties somme et retenue.

4- architecture archi2 of exercice2 is

signal a, b, c, d, e, f: std ulogic;

signal ytemp: std logic;

begin

a <= x1 or x3;

b <= x1 and x3;

c <= x2 and a;

d <= b or c;

```
e <= x1 xor x2;

f <= x3 xor e;

P1: process (d, f, sel)

begin

if sel='0' then

ytemp <= d;

else

ytemp <= 'z';

end if;

end process P1;

P2: process (d, f, sel)

begin

if sel='1' then

ytemp <= f;

else

ytemp <= 'z';

end if;

end process P2;

y <= ytemp;

end architecture archi;
```

Exercice 3 :

```
Library ieee;

use ieee.std_logic_1164.all;
```

```
-- entity statement
Entity Moore_Machine is port
(x, CLK : in  BIT; S1,S2 : out BIT);
End Moore_Machine;
-- architecture statement

Architecture FSM of  Moore_Machine is
type STATE_TYPE  is (Etat0, Etat1, Etat2, Etat3);
signal STATE : STATE_TYPE := Etat0;
signal S :bit_vector (1 downto 0);
begin
--block funtion
NEXT_STATE: block (CLK'event and CLK='1')
begin
-- guarded conditional concurrent signal assignment statement
STATE <= guarded Etat0 when (STATE=Etat0 and x='0') else
                                    Etat1 when (STATE=Etat0 and x='1') else
                                    Etat2 when (STATE=Etat1 and x='1') else
                                    Etat3 when (STATE=Etat2 and x='0') else
                                    Etat2 when (STATE=Etat2 and x='1') else
                                    Etat0 when (STATE=Etat3 and x='0') else
          Etat1 when (STATE=Etat3 and x='1') else
          Etat3 when (STATE=Etat1 and x='0') else
          STATE;
end block NEXT_STATE;
-- unguarded selected concurrent signal assignment statement
```

```
with STATE select
S <= "00" when Etat0,
     "10" when Etat1,
     "11" when Etat2,
     "01" when Etat3,
     "00" when others;



S1<=S(0);
S2<=S(1);

End FSM;
```

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

--library SYNTH ;
--use SYNTH.vhdlsynth.all ;

----- Définition de l'entité----
ENTITY diagramme IS
PORT (raz, clk :IN STD_LOGIC;  -- Ne pas oublier remise à 0 et horloge !
```

```
                x :IN bit;

               S1,S2 :OUT STD_LOGIC);

END diagramme;


------ Définir L'architecture de description d'état ----------------

ARCHITECTURE arch_diagramme OF diagramme IS

        TYPE etat_4 IS (A, B, C, D);      --définir une liste des états...

  SIGNAL etat,etat_suiv :etat_4 := A;  --et 2 signaux: états courant et –-suivant, contenant la valeur d'un état de la liste (initialisée à M0) .

  signal S :std_logic_vector (1 downto 0);

BEGIN

        definir_etat:PROCESS( raz, clk) -- "definir_etat":label optionnel

                BEGIN

                        If raz = '1' THEN

                                etat <= A;                  --Le PACKAGE std_logic_1164

                        ELSIF rising_edge(clk) THEN          --en permet l'utilisation.

                            etat <= etat_suiv ;

                        END IF;                                      --Mise à jour de la variable

        END PROCESS definir_etat;


---------- Définir les états des sorties----------


sorties : process (etat, x)               --Le PROCESS doit contenir une...

          BEGIN                                                  --structure de CAS unique dépendant

                        --de la variable d'état.

CASE etat IS
```

```
  --Le signal de l’état suivant n’est
--attribué que dans la structure CASE

                    WHEN A => S <= "00";

                    If x ='1' then etat_suiv <= B;
                    elsif x ='0' then etat_suiv <= A;
                    end if;

                    WHEN B => S <= "10";

                    If x ='1' then etat_suiv <= C;
                    elsif x ='0' then etat_suiv <= D;
                    end if;

                    WHEN C => S <= "11";

                    If x ='1' then etat_suiv <= C;
                    elsif x ='0' then etat_suiv <= D;
                    end if;

                    WHEN D => S <= "01";

                    If x ='1' then etat_suiv <= B;
                    elsif x ='0' then etat_suiv <= A;
                    end if;
```

```
                    END CASE;

END process sorties ;

S1<=S(0);

S2<=S(1);


END arch_diagramme ;          -- ne pas oublier de terminer l'architecture !
```